Graph Rationalization with Environment-based Augmentations

Gang Liu
University of Notre Dame
Notre Dame, IN, USA
gliu7@nd.edu

Tong Zhao
University of Notre Dame
Notre Dame, IN, USA
tzhao2@nd.edu

Jiaxin Xu
University of Notre Dame
Notre Dame, IN, USA
jxu24@nd.edu

Tengfei Luo
University of Notre Dame
Notre Dame, IN, USA
tluo@nd.edu

Meng Jiang
University of Notre Dame
Notre Dame, IN, USA
mjiang2@nd.edu

ABSTRACT

Rationale is defined as a subset of input features that best explains or supports the prediction by machine learning models. Rationale identification has improved the generalizability and interpretability of neural networks on vision and language data. In graph applications such as molecule and polymer property prediction, identifying representative subgraph structures named as graph rationales plays an essential role in the performance of graph neural networks. Existing graph pooling and/or distribution intervention methods suffer from the lack of examples to learn to identify optimal graph rationales. In this work, we introduce a new augmentation operation called *environment replacement* that automatically creates virtual data examples to improve rationale identification. We propose an efficient framework that performs rationale-environment separation and representation learning on the real and augmented examples in *latent spaces* to avoid the high complexity of explicit graph decoding and encoding. Comparing against recent techniques, experiments on seven molecular and four polymer datasets demonstrate the effectiveness and efficiency of the proposed augmentation-based graph rationalization framework. Data and the implementation of the proposed framework are publicly available[1].

CCS CONCEPTS

• **Applied computing** → **Chemistry**; • **Computing methodologies** → **Learning latent representations**.

KEYWORDS

Graph Learning, Graph Neural Network, Molecule Property, Data Augmentation, Rationalization

ACM Reference Format:
Gang Liu, Tong Zhao, Jiaxin Xu, Tengfei Luo, and Meng Jiang. 2022. Graph Rationalization with Environment-based Augmentations. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3534678.3539347

[1]https://github.com/liugangcode/GREA

KDD '22, August 14–18, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9385-0/22/08…$15.00
https://doi.org/10.1145/3534678.3539347

1 INTRODUCTION

Graph property prediction has attracted attention in different research fields like chemoinformatics and bioinformatics where small molecules are represented as labelled graphs of atoms [7, 9, 48]. Besides, materials informatics for *polymers* has emerged in recent years from property prediction to inverse design [4, 11]. Polymer are materials consisting of macromolecules, composed of many repeating units. They are ubiquitous in applications ranging from plastic cups and electronics to aerospace structures. New engineering and environmental challenges demand that polymers possess unconventional properties such as high-temperature stability, excellent thermal conductivity, and biodegradability [16, 33]. It's important to integrate data science and machine learning into polymer informatics on the tasks of graph classification and regression.

To automate feature extraction from graph data, graph neural network (GNN) models learn node representations through nonlinear functions and layers that aggregate information from node neighborhood [8, 12, 26, 35, 42]. Graph pooling is a central component of the GNN architecture that learns a cluster assignment for nodes and passes cluster nodes and their representations to the next layer [14, 40]. The final layer returns the representations of entire graphs. Despite the advances of various GNN models, the limitation of data size makes them easily fall into *over-fitting and poor generalizability*. For example, the number of graphs in molecule benchmark datasets is usually in the range of 1,000 and 10,000; and the size of polymer datasets is even smaller (e.g., ~600) [17].

Rationalization techniques have been designed to solve the above problem in vision and language data, where the rationale is defined as a subset of input features that best explains or supports the prediction by machine learning models [1, 2, 23]. However, graph rationalization has not been extensively studied, which aims at identifying representative subgraph structures for accurate and interpretable graph property prediction. Related work mainly focused on advancing graph pooling methods, but cluster assignment could not reflect the most essential part that led to accurate prediction [6, 19]. A very recent technique named DIR [34] employed two GNN modules to discover invariant graph rationales: one module separates each input graph into a rationale subgraph and an environment subgraph; the other is a graph property predictor based on the rationale subgraph. As shown at the top in Figure 1, given

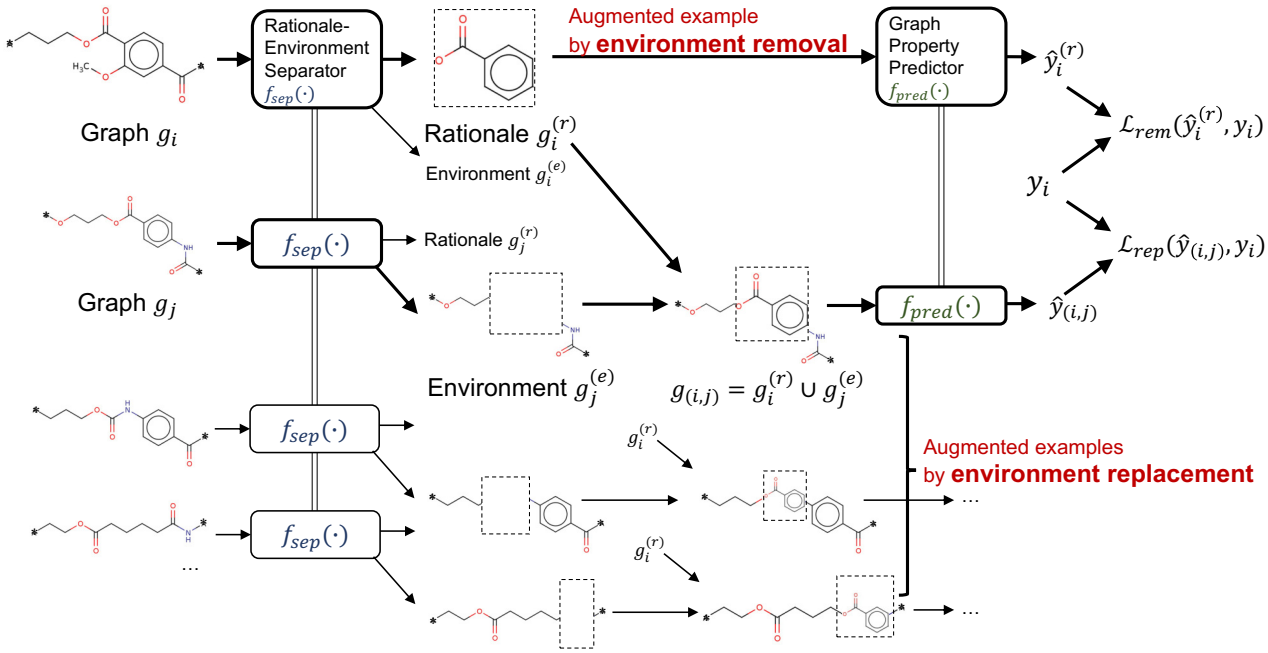


Figure 1: Graph rationalization identifies a rationale subgraph that best explains or supports the prediction of graph property. Our work makes the first attempt to improve graph rationalization by graph data augmentations with *environment subgraphs* which are the remaining parts after rationale identification. It proposes new augmentation operations, designs and develops a novel graph rationalization framework, and conducts experiments on a large set of molecule and polymer data.

graph g_i, the separator f_{sep} identifies rationale $g_i^{(r)}$, and the predictor f_{pred} gives label $\hat{y}_i^{(r)}$ based on the rationale. DIR conducted interventions on training distribution to improve the invariance. Unfortunately, when the data size was small, f_{sep} could hardly find good rationales, as reported in our later experiments.

In this work, we make the first attempt to enhance graph rationalization by graph data augmentations. Existing augmentation methods were mainly heuristic modification of graph structure, which could not directly support the identification of graph rationales [22, 31, 32, 46]. We present two augmentation methods based on *environment subgraphs* that are the remaining parts in the graph after rationale identification. First, rationales are used to train the property predictor, which can be considered as graph examples augmented by *environment removal*. Second, we replace the environment of input graph with the environment of another graph in the batch: to generate an augmented example: this augmentation method is called *environment replacement*. The idea is that the rationale can be accurately identified and/or separated from the input graph when the augmented examples are expected to have the same label of the input graph example.

Figure 1 presents the idea of generating virtual data for small datasets via data augmentations. Suppose we have rationale $g_i^{(r)}$ separated from input graph g_i. We use the same GNN-based separator to find environment subgraph $g_j^{(e)}$ from another graph g_j in the batch. The example augmented by environment replacement is denoted by $g_{(i,j)} = g_i^{(r)} \cup g_j^{(e)}$. The model is trained on this example to predict label $\hat{y}_{i,j}$ to be the same as y_i that is the observed label of g_i. We compute two losses on the augmented examples, $\mathcal{L}_{rem}$ and $\mathcal{L}_{rep}$ ("rem" for removal and "rep" for replacement), and jointly optimize f_{sep} and f_{pred} by their combination.

The key challenge in the idea implementation is the high computational complexity of decoding for *explicit graph forms* of rationales, environment subgraphs, and augmented examples, as well as encoding them for representation learning and property prediction. Moreover, it is scientifically and technically difficult to explicitly combine rationale $g_i^{(r)}$ and environment $g_j^{(e)}$ from different graphs, as shown in the three augmented examples $g_{(i,j)}$ in Figure 1. To address these challenges, we hypothesize that the *contextualized representations of nodes* play a significant role in rationales, environment subgraphs, and augmented graphs. Thus, we create the representations of all these objects from *one latent space*.

In this paper, we propose a novel, efficient framework of Graph Rationalization enhanced by Environment-based Augmentations (GREA). It performs rationale-environment separation and representation learning on the real and augmented examples in one latent space to avoid the high complexity of explicit subgraph decoding and encoding. Figure 2 presents the architecture of GREA with a few steps. First, it employs GNN_1 and MLP_1 models to infer the probability of nodes being classified into rationale subgraph $\mathbf{m}$. Second, it employs GNN_2 to create contextualized node representations $\mathbf{H}$. Then, it *directly* creates the representation vectors of rationales, environment subgraphs and environment-replaced examples, denoted by $\mathbf{h}_i^{(r)}$, $\mathbf{h}_i^{(e)}$, and $\mathbf{h}_{(i,j)}$, respectively. Note that DIR [34] used a GNN to generate a matrix of masks that indicate the importance of edges and then select the top-K edges with the highest masks to construct the rationale. Then it had to run GNNs

on all the explicit graph objects. Instead, our GREA uses **m** and **H** to compute the representation vectors of the artificial graphs.

We conduct experiments on seven molecule and four polymer datasets. Results demonstrate the advantages of GREA over baselines. For example, it significantly reduces the prediction error on oxygen permeability of polymer membrane with only 595 training examples. The oxygen permeability defines how easily oxygen passes through a particular material. Accurate prediction will speed up material discovery for healthcare and energy utilization.

The main contributions of this work are summarized below:

- the first attempt to improve graph rationale identification using data augmentations, including environment replacement, for accurate and interpretable property prediction;
- a novel and efficient framework that performs rationale-environment separation and representation learning on real and augmented examples in one latent space;
- extensive experiments on more than ten molecule and polymer datasets to demonstrate the effectiveness and efficiency of the proposed framework.

2 RELATED WORK

There are four research topics related to the proposed work. We briefly present their recent studies and compare with ours.

2.1 Graph Property Prediction

Learning representations and predicting properties of entire graphs is important for chemistry, biology, and material sciences, where molecule and polymer data can be structured as graphs [9]. When RDKit is widely used to generate molecular fingerprints [13], graph neural networks (GNNs) such as Graph Convolutional Network (GCN) [12], Graph Attention Networks (GAT) [26], and GraphSAGE [8] have automated representation learning with nonlinear functions from graph data [10, 18, 27–30, 35, 42, 43].

In the GNN models, graph pooling is a central component of their architectures as a cluster assignment function to find local patches in graphs [19]. For example, DiffPool presented a differentiable graph pooling module that learned a differentiable soft cluster assignment for nodes at each layer of a deep GNN, mapped nodes to a set of clusters, and then formed the coarsened input for the next GNN layer [40]. Lee et al. proposed self-attention graph convolution that allows graph pooling to consider both node features and graph topology [14]. Gao and Ji proposed graph pooling and unpooling operations in Graph U-Nets [6]. Xu et al. presented a theoretical framework for analyzing the representational power of GNNs through the graph pooling functions [37]. While graph pooling identifies soft clusters that effectively aggregate information from nodes [39], our work identifies representative subgraph structures for accurate and interpretable predictions of GNN models.

2.2 Graph Rationalization

Most rationalization techniques identify the small subset of input features by maximizing the predictive performance based only on the subset itself, called rationale. To rule out spurious correlation between the input features and the output, Chang et al. proposed the concept of invariant rationalization by modeling different environments as non-causal input to train predictors [2]. Rosefeld et al. offered formal guarantees for improvement of the invariant causal prediction on out-of-distribution generalization [1, 23].

By introducing causal modeling into GNN optimization, Fan et al. presented a causal representation framework for GNN models to perform on out-of-distribution graphs [5]. Li et al. proposed OOD-GNN that employed a novel nonlinear graph representation decorrelation method that used random Fourier features to encourage GNNs to eliminate the statistical dependence between relevant and irrelevant graph representations [15]. Very recently, Wu et al. proposed the first work called DIR to approach causal rationales for GNNs to improve the interpretability and predictive performance on out-of-distribution data [34]. DIR conducted interventions on the training distribution to create multiple distributions. Unfortunately, distribution intervention might not be the optimal solution to graph rationale identification. Also, the edge selection method suffers from high computational complexity for rationale creation. Moreover, the studies were mainly performed on synthetic data. In this paper, we make the first attempt to define "environment" in graph data, augment data examples by environment replacement, develop an efficient framework, and conduct experiments on a large set of real molecule and polymer data. We find that augmentation-enhanced graph rationalization is more effective than DIR.

2.3 Graph Data Augmentation

Graph data augmentation (GDA) techniques [3, 44, 45, 47] have improved the performance on semi-supervised node classification, such as DropEdge [22], NodeAug [32], and GAug [46]. Besides, many GDA techniques have been designed for graph-level tasks, aiming at creating new training examples by modifying input graph data examples. For example, GraphCrop regularized GNN models for better generalization by cropping subgraphs or motifs to simulate real-world noise of sub-structure omission [31]. M-Evolve presented two heuristic algorithms including random mapping and motif-similarity mapping to generate weakly labeled data for small datasets [48]. MH-Aug adopted the Metropolis-Hastings algorithm to create augmented graphs from an explicit target distribution for semi-supervised learning [21]. Meanwhile, graph contrastive learning learned unsupervised representations of graphs using graph data augmentations to incorporate various priors [41]. Zhu et al. [49] proposed adaptive augmentation that incorporated various priors for topological and semantic aspects of graphs. Specifically, it designed augmentation schemes based on node centrality measures to highlight important connective structures and corrupted node features by adding noise to unimportant node features. A comprehensive survey of GDA is given by Zhao et al. [44].

2.4 Graph Learning on Polymer Data

Material informatics uses machine learning approaches to fast screen material candidates or generate new materials meeting certain criteria, so as to reduce the time of material development. When most related research performed on molecule data [7], polymer researchers have developed a benchmark database and developed machine learning techniques for polymer data, called polymer embeddings [4, 11]. They can be used to perform several polymer informatics regression tasks for density, glass transition temperature, melting temperature, and dielectric constants [16, 17, 33].

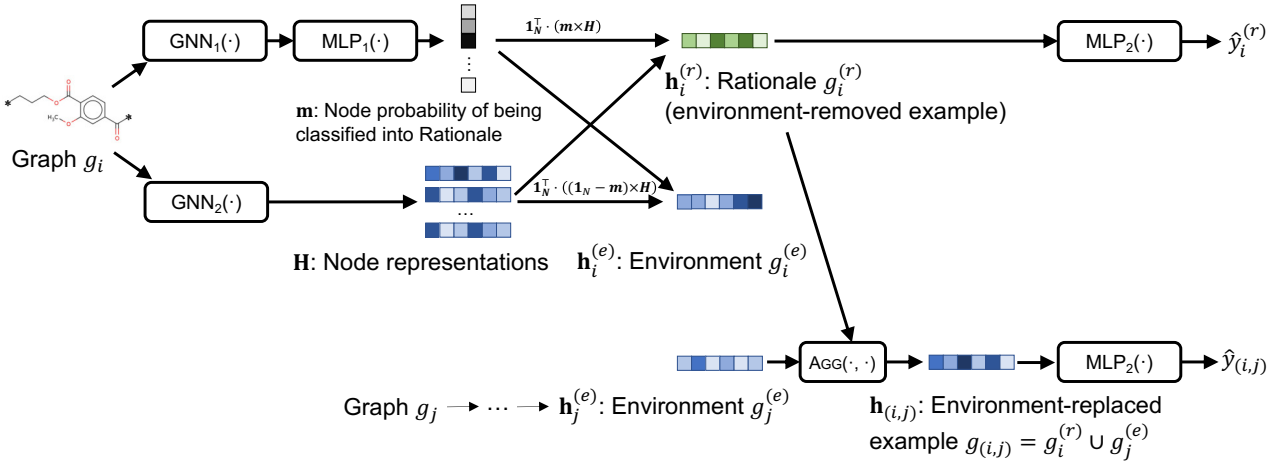


Figure 2: The architecture of the proposed graph rationalization framework: It performs the creation and representation learning of environment-based augmented examples in a *latent space*, instead of decoding every example into a graph form and running a GNN encoder on it. This design aligns graph representation spaces and avoids high computational complexity.

3 PROBLEM DEFINITION

Graph Property Prediction. Let $g = (\mathcal{V}, \mathcal{E})$ be a graph of N nodes and M edges, where $\mathcal{V}$ is the set of nodes (e.g., atoms) and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges (e.g., bonds between atoms). We use $y \in \mathcal{Y}$ to denote the graph-level property of g, where $\mathcal{Y}$ is the value space. It can have a categorical or numerical value, corresponding to the task of classification or regression, respectively.

A graph property predictor f_{pred} takes a graph g as input and predicts its label $\hat{y}$. Specifically, a GNN-based predictor employs a GNN encoder to generate node representations $\mathbf{H}$ from g:

$$\mathbf{H} = \left[\cdots, \vec{h}_v, \cdots\right]_{v\in\mathcal{V}}^{\top} = \text{GNN}(g) \in \mathbb{R}^{N\times d}, \tag{1}$$

where $\vec{h}_v \in \mathbb{R}^d$ is the representation vector of node v in graph g. GNN encoder GNN$(\cdot)$ can be chosen as GCN [12] or GIN [37].

Once the node representations are ready, a multilayer perceptron (MLP) can project them into a one-dimensional space to obtain a scalar for each node as $m_v = \text{MLP}(\vec{h}_v)$. As we are more interested in graph-level classification or regression, we first use a readout operator (e.g., average pooling) to get the graph representation $\mathbf{h}$ and then apply a MLP to project it to a graph label:

$$\mathbf{h} = \text{READOUT}(\mathbf{H}) \in \mathbb{R}^d, \quad \hat{y} = \text{MLP}(\mathbf{h}) \in \mathcal{Y}. \tag{2}$$

Graph Rationalization. Following the existing literature on graph rationalization [5, 6, 14, 34, 40] and GNN explanation [39], we use rationale $g^{(r)} = (\mathcal{V}^{(r)}, \mathcal{E}^{(r)})$ to indicate the causal subgraph of the property y, where $g^{(r)}$ is a subgraph of g such that $\mathcal{V}^{(r)} \subseteq \mathcal{V}$ and $\mathcal{E}^{(r)} \subseteq \mathcal{E}$. We use $g^{(e)}$ to denote the environment subgraph, which is the complementary subgraph of $g^{(r)}$ in g. In contrast with the rationale subgraph $g^{(r)}$, the environment subgraph $g^{(e)}$ corresponds to the non-causal part of the graph data, which has no causal relationship with the target graph property [2, 34].

Let f_{sep} be a GNN-based graph rationalization model that splits an input graph g into a rationale subgraph $g^{(r)}$ and an environment subgraph $g^{(e)}$. Existing graph rationalization methods used only the rationale subgraph as input for property prediction [6, 14, 34, 40]:

$$\hat{y} = \hat{y}^{(r)} = f_{pred}(g^{(r)}), \tag{3}$$

where $f_{pred}(\cdot) = \text{MLP}(\text{READOUT}(\text{GNN}(\cdot)))$ and $\hat{y}^{(r)}$ denotes the predicted property of the rationale subgraph $g^{(r)}$.

Unfortunately, when suffering from lack of training examples, these methods chose to discard environment subgraphs at the training stage. In the next section, we present a novel framework showing our idea that environment subgraphs can provide natural noise through data augmentation to improve graph rationalization.

4 PROPOSED FRAMEWORK

In this section, we introduce a novel graph rationalization framework GREA. The key idea is to augment the rationale subgraph by removing its own environment subgraph and/or combining it with different environment subgraphs. Figure 2 shows the overall architecture of GREA: GNN_1 and MLP_1 first separate input graph g into rationale subgraph $g^{(r)}$ and environment subgraph $g^{(e)}$; GNN_2 next generates node representations $\mathbf{H}$ using Eq.(1); the rationale subgraph's representation $\mathbf{h}_i^{(r)}$ is then combined with different environment subgraph's representations $\mathbf{h}_j^{(e)}$ for the augmented graph's representations $\mathbf{h}_{(i,j)}$; finally, both $\mathbf{h}_i^{(r)}$ and $\mathbf{h}_{(i,j)}$ are fed into MLP_2 for the prediction of y_i during training as Eq.(2).

4.1 Rationale-Environment Separation

To separate input graph g into rationale subgraph $g^{(r)}$ and environment subgraph $g^{(e)}$, the rationale-environment separator consists of two components: a GNN encoder (GNN_1) that generates latent node representations and a MLP decoder (MLP_1) that maps the node representations to a mask vector $\mathbf{m} \in (0, 1)^N$ on the nodes in the set $\mathcal{V}$. $m_v = Pr(v \in \mathcal{V}^{(r)})$ is the node-level mask that indicates the probability of node $v \in \mathcal{V}$ being classified into the rationale subgraph. The mask can be on either a node or an edge [34]. we choose to learn masks on the nodes to avoid the computational

complexity of edge selection. Hence, $\mathbf{m}$ can be calculated as

$$\mathbf{m} = \sigma(\text{MLP}_1(\text{GNN}_1(g))), \tag{4}$$

where σ denotes the sigmoid function. Based on $\mathbf{m}$, we have $(\mathbf{1}_N - \mathbf{m})$ that indicates the probability of nodes being classified into the environment subgraph. GNN_1 and MLP_1 make up the GNN-based graph rationalization model f_{sep} mentioned in Section 3.

GREA uses another GNN encoder to generate contextualized node representations $\mathbf{H}$: $\mathbf{H} = \text{GNN}_2(g)$. With $\mathbf{m}$ and $\mathbf{H}$, the rationale subgraph and environment subgraph can be easily separated in the *latent space*. Using sum pooling, we have

$$\mathbf{h}^{(r)} = \mathbf{1}_N^\top \cdot (\mathbf{m} \times \mathbf{H}), \quad \mathbf{h}^{(e)} = \mathbf{1}_N^\top \cdot ((\mathbf{1}_N - \mathbf{m}) \times \mathbf{H}), \tag{5}$$

where $\mathbf{1}_N$ denotes the N-size column vector with all entries as 1, and $\mathbf{h}^{(r)}, \mathbf{h}^{(e)} \in \mathbb{R}^d$ are the representation vectors of graph $g^{(r)}$ and $g^{(e)}$, respectively.

4.2 Environment-based Augmentations

Suppose $g_1, g_2, \ldots, g_B$ are the input graphs in one batch for training, where B is known as batch size. The rationale-environment separator has generated the graph representations of rationale and environment subgraphs for each graph g_i. That is, we have $\{(\mathbf{h}_1^{(r)}, \mathbf{h}_1^{(e)}), (\mathbf{h}_2^{(r)}, \mathbf{h}_2^{(e)}), \ldots, (\mathbf{h}_B^{(r)}, \mathbf{h}_B^{(e)})\}$. We design environment-based augmentations in the latent space of graph representations.

4.2.1 Environment Removal Augmentation. As graph rationalization aims to find the rationale subgraph which is regarded as the causal factor of graph property, the rationale itself should be good for property prediction. As in the graph pooling methods [6, 14] and the graph rationalization as defined in Eq. (3), the environment removal augmentation uses the rationale subgraph only for training the graph property predictor. That is, given the rationale subgraph representation $\mathbf{h}_i^{(r)}$ of graph g_i, the predicted label is

$$\hat{y}_i^{(r)} = \text{MLP}_2\big(\mathbf{h}_i^{(r)}\big). \tag{6}$$

4.2.2 Environment Replacement Augmentation. As aforementioned in Section 3, the environment subgraphs can be viewed as natural noises on the rationale subgraphs. Hence, in order to enhance the model's robustness against the noise signal brought by the environment subgraphs, for each graph g_i, we combine its rationale subgraph $g_i^{(r)}$ not only with its own environment subgraph $g_i^{(e)}$, but also with all other environment subgraphs $g_j^{(e)}$, $j \in \{1, 2, \ldots, B\} \setminus \{i\}$ in the batch. By replacing the environment subgraph with other environment subgraphs in the batch, the environment replacement augmentation generates $B - 1$ augmented data samples for each graph during training. As the environment replacement happens on the latent space, an aggregation function $\text{AGG}(\cdot, \cdot)$ is used to combine the rationale subgraph representation $\mathbf{h}_i^{(r)}$ and environment subgraph representation $\mathbf{h}_j^{(e)}$. The aggregation function can be any combining/pooling functions such as concatenation, sum pooling, and max pooling. Taking the element-wise sum pooling as an example, the graph representation $\mathbf{h}_{(i,j)}$ of a combined graph of rationale subgraph $g_i^{(r)}$ and environment subgraph $g_j^{(e)}$ can be calculated as below:

$$\mathbf{h}_{(i,j)} = \text{AGG}\big(\mathbf{h}_i^{(r)}, \mathbf{h}_j^{(e)}\big) = \mathbf{h}_i^{(r)} + \mathbf{h}_j^{(e)}. \tag{7}$$

Table 1: Statistics of eleven datasets for graph property prediction: The four top rows are polymer datasets. The prediction tasks are graph regression. The seven bottom rows are molecule datasets. Their tasks are graph classification.

Dataset	# Graphs	Avg./Max # Nodes	Avg./Max # Edges
GlassTemp	7,174	36.7 / 166	79.3 / 362
MeltingTemp	3,651	26.9 / 102	55.4 / 212
PolyDensity	1,694	27.3 / 93	57.6 / 210
O_2Perm	595	37.3 / 103	82.1 / 234
ogbg-HIV	41,127	25.5 / 222	54.9 / 502
ogbg-ToxCast	8,576	18.8 / 124	38.5 / 268
ogbg-Tox21	7,831	18.6 / 132	38.6 / 290
ogbg-BBBP	2,039	24.1 / 132	51.9 / 290
ogbg-BACE	1,513	34.1 / 97	73.7 / 202
ogbg-ClinTox	1,477	26.2 / 136	55.8 / 286
ogbg-SIDER	1,427	33.6 / 492	70.7 / 1010

For the graph representations $\mathbf{h}_{(i,j)}$ generated by the environment replacement augmentation, the MLP property predictor is trained to predict y_i. That is,

$$\hat{y}_{(i,j)} = \text{MLP}_2\big(\mathbf{h}_{(i,j)}\big). \tag{8}$$

The graph representations generated by both environment removal augmentation and environment replacement augmentation (i.e., $\mathbf{h}_i^{(r)}$ and $\mathbf{h}_{(i,j)}$) are fed into the same property predictor MLP_2. The GNN-based property predictor f_{pred} defined in Section 3 includes MLP_2 and GNN_2 that generates the contextualized node representation $\mathbf{H}$.

4.2.3 Optimization. During training, the type of loss function on the observed graph property (y_i) and predicted labels ($\hat{y}_i^{(r)}$ and $\hat{y}_{(i,j)}$) depends on the type of the property label. For example, when the graph property y has binary values in the binary classification task, we use the standard binary cross-entropy loss. When the graph property y has real values in the graph regression task, we use the mean squared error (MSE) loss. Without loss of generality, suppose we focus on the binary classification task. Given a batch of B graphs $g_1, g_2, \ldots, g_B$, the loss functions for each graph example g_i and its label y_i are defined as

$$\mathcal{L}_{rem} = y_i \cdot \log \hat{y}_i^{(r)} + (1 - y_i) \cdot \log\big(1 - \hat{y}_i^{(r)}\big), \tag{9}$$

$$\mathcal{L}_{rep} = \frac{1}{B} \sum_{j=1}^{B} \big(y_i \cdot \log \hat{y}_{(i,j)} + (1 - y_i) \cdot \log(1 - \hat{y}_{(i,j)})\big), \tag{10}$$

where $\mathcal{L}_{rem}$ is the loss for the examples created by environment removal augmentation, and $\mathcal{L}_{rep}$ is the loss for the examples created by the environment replacement augmentation.

Moreover, the following regularization term is used to control the size of the selected rationale subgraph:

$$\mathcal{L}_{reg} = \Big(\frac{\mathbf{1}_N^\top \mathbf{m}}{N} - \gamma\Big) + \Big(\frac{\sum_{k:\mathbf{m}_k>0} 1}{N} - \gamma\Big), \tag{11}$$

where $\gamma \in [0, 1]$ is a hyperparamter to control the expected size of the rationale subgraph $g^{(r)}$. The node probabilities classified as the rationale should only exist on certain nodes with the expected size. Therefore, we use the first term to penalize the number of nodes

in the rationale when it deviates from our expectations and the second term to encourage an uneven distribution for $\mathbf{m}$.

We use the alternate training schema in Chang et al. [2] to train GREA. That is, we iteratively train f_{sep} (GNN_1 and MLP_1) and f_{pred} (GNN_2 and MLP_2) for a fixed number of epochs T_{sep} and T_{pred}, respectively. The loss functions for training GREA are

$$\mathcal{L}_{pred} = \mathcal{L}_{rem} + \alpha \cdot \mathcal{L}_{rep}, \tag{12}$$

$$\mathcal{L}_{sep} = \mathcal{L}_{rem} + \alpha \cdot \mathcal{L}_{rep} + \beta \cdot \mathcal{L}_{reg}, \tag{13}$$

where $\mathcal{L}_{pred}$ in Eq. (12) and $\mathcal{L}_{sep}$ in Eq. (13) are used to train f_{sep} (GNN_1 and MLP_1) and f_{pred} (GNN_2 and MLP_2), respectively. α and β are hyperparameters that control the weights of $\mathcal{L}_{rep}$ and $\mathcal{L}_{reg}$, respectively. During inference, $\hat{y}_i^{(r)}$ is used as the final predicted property of input graph g_i.

5 EXPERIMENTS

We conduct experiments to answer the following questions:

- **Q1)** Effectiveness: Does the proposed GREA make more accurate prediction on molecule and polymer properties than existing graph classification/regression methods?
- **Q2)** Ablation study: Do the environment-based augmentations make positive effect on the performance?
- **Q3)** Case study: Based on domain expertise, are the polymer rationale examples identified by GREA representative?
- **Q4)** Efficiency: Does the *latent space-based design* for augmentations perform faster than explicit graph decoding and encoding? Can we empirically analyze the complexity?
- **Q5)** Sensitivity analysis: Is the performance of GREA sensitive to hyperparameters such as α, β, and AGG($\cdot$)?

5.1 Experimental Settings

5.1.1 Datasets. We conduct experiments on **four** polymer datasets and **seven** molecule datasets. The statistics of the datasets are given in Table 1, such as number of graphs and average size of graphs. The four datasets GlassTemp, MeltingTemp, PolyDensity, and O_2Perm are used to predict different properties of polymers such as *glass transition temperature* (°C), *polymer density* g/cm^3, *melting temperature* (°C), and *oxygen permeability* (Barrer). For all the polymer datasets, we randomly split by 60%/10%/30% for training, validation, and test. Besides polymer datasets, we use seven molecule datasets from the graph property prediction task on Open Graph Benchmark or known as OGBG. For all molecule datasets, we use the scaffold splitting procedure as OGBG adopted [9]. It attempts to separate structurally different molecules into different subsets, which provides a more realistic estimate of model performance in experiments [36]. Dataset descriptions with details are presented in the Appendix A.

5.1.2 Evaluation Metrics. On the polymer datasets, we perform the tasks of graph regression. We use the coefficient of determination (R^2) and Root Mean Square Error (RMSE) as evaluation metrics according to previous works [9, 17]. On the molecule datasets, we perform the tasks of graph binary classification using the Area under the ROC curve (AUC) as the metric. To evaluate model efficiency, we use the computational time per training batch (in seconds).

5.1.3 Baseline Methods. There are three categories of related methods that we can compare GREA with. The first category is *graph pooling* methods that aim at finding (soft) cluster assignment of nodes towards aggregated representations of graph. They are U-NetsPool [6] and SelfAttnPool [14]. The second category improves the *optimization and generalization* of learned representations. They include StableGNN [5], OOD-GNN [15], and IRM [1]. The third is DIR for *graph rationale identification* that was proposed in a very recent work by Wu et al. [34]. To investigate the effect of *environment replacement augmentation* (denoted by RepAug as a module that may be used or not in the methods), we implement two method variants: (1) DIR+RepAug: We add environment-replaced augmentation to DIR [34] to identify rationales, however, it has to explicitly decode and encode the rationales; (2) GREA−RepAug: We disable the environment replacement augmentation and use only the environment removal augmentation, i.e., rationale subgraphs in GREA. In the experiments, we study two types of GNN models (GCN [12] and GIN [37]) as graph encoders for all the methods. Please refer to Appendix B for details of implementation.

5.2 Results on Effectiveness (Q1)

Table 2 presents the results on polymer property regression with R^2 and RMSE metrics. Table 3 presents the results on molecule property classification using AUC. Underlined are for the best baseline(s). The best baseline is OOD-GNN for its elimination of the statistical dependence between property-relevant graph representation and property-irrelevant graph representation. The first graph rationalization method DIR was evaluated on synthetic data [34]; unfortunately, it performs poorly on real polymer and molecule datasets because it selects edges to create rationale subgraphs and thus loses the original contextual information of atoms in the the rationale representations. Compared to them, our GREA with either GCN or GIN consistently achieves the best performance on all the polymer and molecule datasets. On the PolyDensity dataset, GREA with GCN improves R^2 over OOD-GNN relatively by +3.91%. On MeltingTemp, GREA with GIN produces 1.56× R^2 over DIR.

5.3 Ablation Study on GREA (Q2)

Tables 2 and 3 have presented the results of DIR+RepAug and GREA−RepAug. DIR+RepAug is a variant of baseline method DIR by enabling *environment replacement augmentations* for training. GREA−RepAug is a variant of our GREA that disables the replacement augmentations and uses *environment removal* only for training. Clearly, DIR+RepAug outperforms DIR, showing positive effect of the replacement augmentations. And the performance of GREA−RepAug is not satisfactory. Environment replacement augmentations are effective for training graph rationalization methods.

5.4 Case Study on Polymer Data (Q3)

Given test polymer examples in the O_2Perm dataset, we visualize and compare the rationale subgraphs that are identified by from DIR [34] and our GREA in Figure 3. We have three observations.

First, the rationales identified by GREA have more *coherent structures of atom nodes* than those identified by DIR. The red boxes show that quite a few edges in the rationales by DIR are far separated. This is because DIR explicitly decodes the subgraphs by selecting edges. Our GREA estimates the probability of *nodes* being

Table 2: Results on polymer property prediction: GREA consistently achieves the highest R^2 and smallest RMSE.

		GlassTemp		MeltingTemp		PolyDensity		O_2Perm	
		R^2 ↑	RMSE ↓	R^2 ↑	RMSE ↓	R^2 ↑	RMSE ↓	R^2 ↑	RMSE ↓
GCN [12] as encoder	U-NetsPool [6]	0.839±0.005	44.9±0.7	0.685±0.012	63.4±1.2	0.615±0.053	0.100±0.007	0.833±0.084	865±214
	SelfAttnPool [14]	0.848±0.007	43.5±1.0	0.709±0.008	61.0±0.9	0.688±0.019	0.090±0.003	0.656±0.135	1251±266
	StableGNN [5]	0.809±0.013	48.8±1.6	0.635±0.033	70.0±4.5	0.667±0.070	0.093±0.009	0.676±0.127	1219±241
	OOD-GNN [15]	0.852±0.006	43.0±0.9	0.714±0.025	60.4±2.6	0.676±0.010	0.092±0.001	0.921±0.059	576±212
	IRM [1]	0.830±0.008	46.1±1.1	0.677±0.006	64.2±0.6	0.690±0.016	0.090±0.002	0.871±0.043	770±141
	DIR [34]	0.697±0.061	61.2±6.0	0.380±0.214	87.8±14.	0.656±0.036	0.094±0.005	0.135±0.068	2028±80
	DIR+RepAug	0.800±0.006	56.5±3.2	0.520±0.101	77.8±8.2	0.671±0.033	0.092±0.005	0.915±0.031	626±115
	GREA−RepAug	0.685±0.172	60.6±16.5	0.679±0.034	64.0±3.3	0.686±0.007	0.090±0.001	0.459±0.254	1556±395
	GREA (ours)	**0.855**±0.003	**42.6**±0.5	**0.716**±0.016	**60.2**±1.6	**0.717**±0.023	**0.086**±0.003	**0.941**±0.018	**524**±91
GIN [37] as encoder	U-NetsPool [6]	0.852±0.006	42.9±0.9	0.703±0.009	61.6±0.9	0.635±0.029	0.097±0.004	0.868±0.085	753±250
	SelfAttnPool [14]	0.848±0.003	43.5±0.4	0.726±0.009	59.2±1.0	0.654±0.024	0.095±0.003	0.601±0.267	1265±546
	StableGNN [5]	0.794±0.007	50.8±0.9	0.535±0.061	76.9±5.0	0.642±0.045	0.096±0.006	0.501±0.266	1487±404
	OOD-GNN [15]	0.862±0.007	41.6±1.1	0.721±0.006	59.7±0.6	0.666±0.025	0.093±0.003	0.917±0.029	620±109
	IRM [1]	0.842±0.004	44.5±0.5	0.681±0.008	63.8±0.8	0.682±0.031	0.091±0.004	0.890±0.042	709±146
	DIR [34]	0.594±0.070	71.0±6.0	0.287±0.121	95.1±7.9	0.617±0.045	0.099±0.006	0.501±0.309	1446±537
	DIR+RepAug	0.744±0.029	56.4±3.2	0.542±0.083	76.2±7.0	0.647±0.058	0.095±0.008	0.743±0.150	1054±338
	GREA−RepAug	0.494±0.110	79.0±9.3	0.660±0.107	65.2±9.5	0.717±0.022	0.086±0.003	0.400±0.286	1623±474
	GREA (ours)	**0.864**±0.005	**41.2**±0.8	**0.736**±0.012	**58.0**±1.2	**0.723**±0.030	**0.085**±0.005	**0.930**±0.020	**569**±86

Table 3: Results on molecule property prediction: GREA consistently achieves the highest AUC (↑).

		ogbg-HIV	ogbg-ToxCast	ogbg-Tox21	ogbg-BBBP	ogbg-BACE	ogbg-ClinTox	ogbg-SIDER
GCN [12] as encoder	U-NetsPool [6]	0.7527±0.0104	0.6507±0.0086	0.7492±0.0093	0.6709±0.0176	0.7757±0.0173	0.8450±0.0403	0.6181±0.0121
	SelfAttnPool [14]	0.7733±0.0187	0.6510±0.0076	0.7563±0.0080	0.6602±0.0220	0.7383±0.0541	0.8291±0.0791	0.5718±0.0219
	StableGNN [5]	0.7218±0.0099	0.6520±0.0109	0.7454±0.0059	0.6552±0.0184	0.6607±0.0500	0.7681±0.0778	0.5644±0.0274
	OOD-GNN [15]	0.7580±0.0176	0.6613±0.0046	0.7673±0.0109	0.6795±0.0165	0.8096±0.0132	0.8874±0.0143	0.6133±0.0095
	IRM [1]	0.7702±0.0107	0.6599±0.0063	0.7654±0.0072	0.6892±0.0053	0.7947±0.0186	0.8819±0.0231	0.6035±0.0195
	DIR [34]	0.7466±0.0093	0.5954±0.0154	0.4727±0.0129	0.6559±0.0298	0.6751±0.0323	0.6251±0.0956	0.5331±0.0216
	DIR+RepAug	0.7494±0.0225	0.6632±0.0098	0.7437±0.0054	0.6630±0.0118	0.7677±0.0226	0.8606±0.0144	0.5934±0.0170
	GREA−RepAug	0.7377±0.0210	0.6614±0.0048	0.7808±0.0061	0.6736±0.0077	0.7655±0.0529	0.8708±0.0514	0.6222±0.0166
	GREA (ours)	**0.7794**±0.0065	**0.6662**±0.0041	**0.7822**±0.0093	**0.6986**±0.0175	**0.8191**±0.0240	**0.8961**±0.0150	**0.6316**±0.0151
GIN [37] as encoder	U-NetsPool [6]	0.7375±0.0362	0.6524±0.0126	0.7560±0.0093	0.6809±0.0163	0.8026±0.0105	0.8146±0.0703	0.5929±0.0114
	SelfAttnPool [14]	0.7533±0.0247	0.6351±0.0137	0.7507±0.0110	0.6624±0.0167	0.7348±0.0194	0.7912±0.0995	0.5702±0.0137
	StableGNN [5]	0.7218±0.0078	0.6485±0.0025	0.7381±0.0123	0.6695±0.0120	0.7229±0.0122	0.8559±0.0224	0.5593±0.0172
	OOD-GNN [15]	0.7799±0.0078	0.6697±0.0051	0.7646±0.0038	0.6710±0.0188	0.7800±0.0228	0.8416±0.0496	0.5916±0.0169
	IRM [1]	0.7817±0.0120	0.6641±0.0065	0.7542±0.0084	0.6835±0.0071	0.7977±0.0208	0.8485±0.0215	0.5778±0.0206
	DIR [34]	0.7533±0.0117	0.5927±0.0097	0.5078±0.0313	0.5843±0.0443	0.6115±0.0587	0.6911±0.0810	0.5406±0.0127
	DIR+RepAug	0.7725±0.0249	0.6454±0.0061	0.7453±0.0080	0.6813±0.0203	0.7590±0.0642	0.8561±0.0159	0.5730±0.0115
	GREA−RepAug	0.7770±0.0178	0.6681±0.0066	0.7690±0.0117	0.6737±0.0235	0.7997±0.0380	0.8574±0.0442	0.5988±0.0169
	GREA (ours)	**0.7932**±0.0092	**0.6750**±0.0067	**0.7723**±0.0119	**0.6970**±0.0128	**0.8237**±0.0237	**0.8789**±0.0368	**0.6014**±0.0204

Table 4: Effect of $\text{AGG}(\mathbf{h}_i^{(r)}, \mathbf{h}_j^{(e)})$ in Eq. (7). We use Sum Pooling by default because it generally performs the best.

	MeltingTemp (R^2)	O_2Perm (R^2)	ogbg-HIV (AUC)
Sum Pooling	**0.7362**±0.0115	**0.9304**±0.0202	**0.7932**±0.0092
Mean Pooling	0.7328±0.0068	0.9288±0.0331	0.7810±0.0117
Max Pooling	0.7164±0.0094	0.8984±0.0494	0.7809±0.0137
Concatenation	0.7145±0.0127	0.9240±0.0143	0.7771±0.0096

included in the rationales and uses the *contextualized representations* of atoms in the input graphs to create the representations of rationales. So the rationales have coherent structures of nodes.

Second, the rationales from GREA are *more interpretable and beneficial* than the ones from DIR, based on domain expertise in polymer science. Take a look at the first polymer example in Figure 3. The rationale from GREA includes non-aromatic rings and methyl groups. The former group allows larger free volume elements and lower densities (i.e., enlarge microporousity) in the polymer's repeating units, which positively contributes to the gas permeability [24, 38]. The latter group is hydrophobic and contributes to steric frustration between polymer chains [38], inducing a positive correlation to the permeability. On the other hand, the rationale from DIR would make property predictor overestimate the oxygen permeability, because it suggests that the double-bonded oxygens, ethers, and

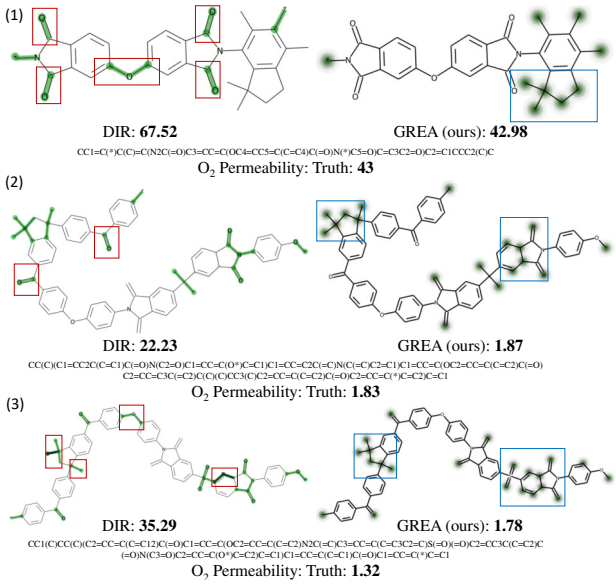
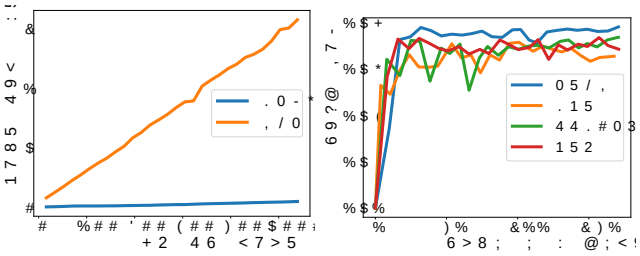


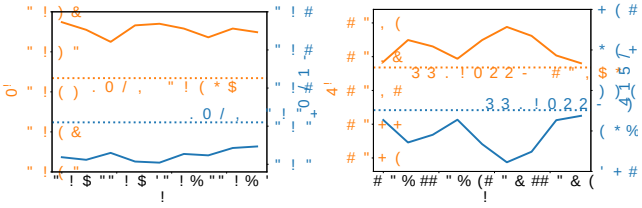
Figure 3: Three polymer examples in O_2Perm test set to compare graph rationales and property predictions by DIR [34] and our GREA. DIR selects *edges* to decode rationale subgraphs. Our GREA estimates the probability of *nodes* being classified into rationales in latent space. The red boxes indicate incoherent edges that DIR selects. The blue boxes indicate coherent node sets that contribute to accurate predictions on oxygen permeability of polymer membrane.



(a) Our GREA runs much faster than DIR when batch size (# graphs) increases.

(b) GREA spends comparable amount of training time to deliver the highest AUC.

Figure 4: Efficiency analysis on the ogbg-HIV dataset.

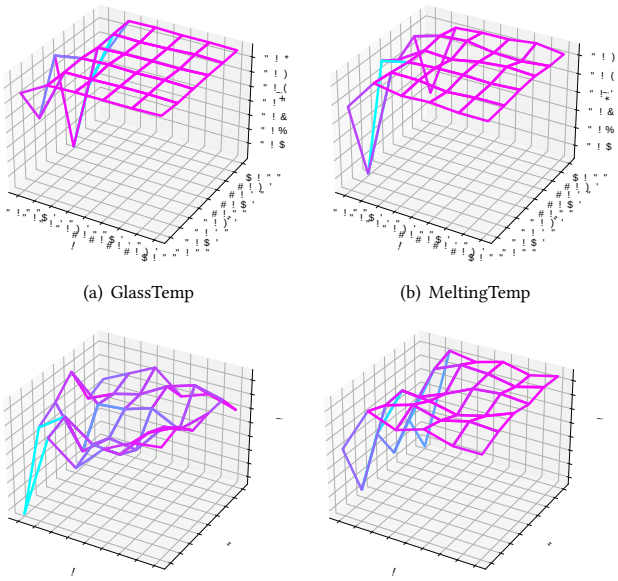


(a) PolyDensity

(b) O_2Perm

Figure 5: On two polymer datasets, the performance of GREA is *not* sensitive to rationale size γ with wide ranges for tuning.

nitrogen atoms are positively correlated with the property. However, it conflicts with observations and conclusions from chemical



(a) GlassTemp

(b) MeltingTemp

(c) PolyDensity

(d) O_2Perm

Figure 6: On four polymer datasets, the performance of GREA (in R^2) is *not* sensitive to hyperparameters α and β in Eq. (13).

experiments in previous literature [38] where researchers argue that the double-bonded oxygens, ethers, and nitrogen atoms are negatively correlated with gas permeability. For the second and third examples, DIR also predicts through double-bonded oxygens, ethers, and nitrogen atoms, and it overestimates the permeability. Our GREA realizes and employs the true relationship between the functional groups and property and successfully suppresses the representations of non-aromatic rings and methyl groups in the prediction. GREA intrinsically discovers correct relationships between rationale subgraphs and the property.

Third, the rationales from GREA are *commonly observed across different polymers*. We expect rationales to have universal indication on the polymer properties. The rationales identified in the second and third examples both have the fused heterocyclic rings (at the right end of the monomers and highlighted by blue boxes).

5.5 Results on Efficiency (Q4)

We conduct efficiency analysis using the ogbg-HIV dataset without losing the generality. Results are presented in Figure 4. When batch size increases, in other words, when a batch has more and more graphs, the time cost per batch of DIR increases significantly; our proposed GREA spends much less time than DIR. Empirically we show that our GREA is more efficient than DIR. This is because GREA does not explicitly decode or encode the subgraphs but directly creates their representations in latent space. Figure 4(b) shows that compared to three most competitive baselines, GREA delivers the highest AUC by learning augmented examples, while spending comparable amount of time.

5.6 Sensitivity Analysis (Q5)

Without losing the generality, we conduct three series of sensitivity analyses. First, Figure 6 shows that on four polymer datasets, the

performance of GREA in terms of R^2 is insensitive to the hyper-parameters α and β in Eq. (13). Second, Figure 5 shows that the performance is insensitive to rationale size γ in Eq. (11). Third, on two polymer datasets and one of the most popular molecule datasets, Table 4 compares the effects of different choices of AGG$(\cdot)$ function that aggregates the representations of rationale and environment subgraphs. Sum pooling is generally the best choice.

6 CONCLUSIONS

In this work, we made the first attempt to improve graph rationale identification using data augmentations, including environment replacement, for accurate and interpretable graph property prediction. We proposed an efficient framework that performs rationale-environment separation and representation learning on real and augmented examples in one latent space. Experiments on molecule and polymer datasets demonstrated its effectiveness and efficiency.

ACKNOWLEDGMENTS

This research was supported in part by NSF Grants IIS-1849816, IIS-2142827, IIS-2146761, and CBET-2102592.

REFERENCES

[1] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. 2019. Invariant risk minimization. In *arXiv:1907.02893*.
[2] Shiyu Chang, Yang Zhang, Mo Yu, and Tommi Jaakkola. 2020. Invariant rationalization. In *ICML*. 1448–1458.
[3] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. 2020. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *AAAI*, Vol. 34. 3438–3445.
[4] Lihua Chen, Ghanshyam Pilania, Rohit Batra, Tran Doan Huan, Chiho Kim, Christopher Kuenneth, and Rampi Ramprasad. 2021. Polymer informatics: Current status and critical next steps. *Materials Science and Engineering: R: Reports* 144 (2021), 100595.
[5] Shaohua Fan, Xiao Wang, Chuan Shi, Peng Cui, and Bai Wang. 2021. Generalizing Graph Neural Networks on Out-Of-Distribution Graphs. In *arXiv:2111.10657*.
[6] Hongyang Gao and Shuiwang Ji. 2021. Graph U-Nets. *IEEE TPAMI* (2021).
[7] Zhichun Guo, Chuxu Zhang, Wenhao Yu, John Herr, Olaf Wiest, Meng Jiang, and Nitesh V Chawla. 2021. Few-Shot Graph Learning for Molecular Property Prediction. In *WWW*. 2559–2567.
[8] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*. 1025–1035.
[9] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open Graph Benchmark: Datasets for Machine Learning on Graphs. In *NeurIPS*.
[10] Meng Jiang, Taeho Jung, Ryan Karl, and Tong Zhao. 2022. Federated Dynamic Graph Neural Networks with Secure Aggregation for Video-based Distributed Surveillance. *TIST* 13, 4 (2022), 1–23.
[11] Chiho Kim, Anand Chandrasekaran, Tran Doan Huan, Deya Das, and Rampi Ramprasad. 2018. Polymer genome: a data-powered polymer informatics platform for property predictions. *The Journal of Physical Chemistry C* 122, 31 (2018), 17575–17585.
[12] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.
[13] Greg Landrum. 2013. RDKit: A software suite for cheminformatics, computational chemistry, and predictive modeling.
[14] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. 2019. Self-attention graph pooling. In *ICML*. 3734–3743.
[15] Haoyang Li, Xin Wang, Ziwei Zhang, and Wenwu Zhu. 2021. OOD-GNN: Out-of-Distribution Generalized Graph Neural Network. In *arXiv:2112.03806*.
[16] Ruimin Ma, Zeyu Liu, Quanwei Zhang, Zhiyu Liu, and Tengfei Luo. 2019. Evaluating polymer representations via quantifying structure–property relationships. *Journal of chemical information and modeling* 59, 7 (2019), 3110–3119.
[17] Ruimin Ma and Tengfei Luo. 2020. PI1M: a benchmark database for polymer informatics. *Journal of Chemical Information and Modeling* 60, 10 (2020), 4684.
[18] Yao Ma, Xiaorui Liu, Tong Zhao, Yozen Liu, Jiliang Tang, and Neil Shah. 2021. A unified view on graph neural networks as graph signal denoising. In *CIKM*. 1202–1211.
[19] Diego Mesquita, Amauri Souza, and Samuel Kaski. 2020. Rethinking pooling in graph neural networks. In *NeurIPS*.
[20] Shingo Otsuka, Isao Kuwajima, Junko Hosoya, Yibin Xu, and Masayoshi Yamazaki. 2011. PoLyInfo: Polymer database for polymeric materials design. In *International Conference on Emerging Intelligent Data and Web Technologies*. 22.
[21] Hyeonjin Park, Seunghun Lee, Sihyeon Kim, Jinyoung Park, Jisu Jeong, Kyung-Min Kim, Jung-Woo Ha, and Hyunwoo J Kim. 2021. Metropolis-Hastings Data Augmentation for Graph Neural Networks. In *NeurIPS*.
[22] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2019. DropEdge: Towards Deep Graph Convolutional Networks on Node Classification. In *ICLR*.
[23] Elan Rosenfeld, Pradeep Kumar Ravikumar, and Andrej Risteski. 2021. The Risks of Invariant Risk Minimization. In *ICLR*.
[24] David F Sanders, Zachary P Smith, Ruilan Guo, Lloyd M Robeson, James E McGrath, Donald R Paul, and Benny D Freeman. 2013. Energy-efficient polymeric gas separation membranes for a sustainable future: A review. *Polymer* 54, 18 (2013), 4729–4761.
[25] A Thornton, L Robeson, B Freeman, and D Uhlmann. 2012. Polymer Gas Separation Membrane Database.
[26] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.
[27] Daheng Wang, Meng Jiang, Munira Syed, Oliver Conway, Vishal Juneja, Sriram Subramanian, and Nitesh V Chawla. 2020. Calendar graph neural networks for modeling time structures in spatiotemporal user behaviors. In *KDD*. 2581–2589.
[28] Daheng Wang, Zhihan Zhang, Yihong Ma, Tong Zhao, Tianwen Jiang, Nitesh Chawla, and Meng Jiang. 2021. Modeling co-evolution of attributed and structural information in graph sequence. *IEEE TKDE* (2021).
[29] Daheng Wang, Zhihan Zhang, Yihong Ma, Tong Zhao, Tianwen Jiang, Nitesh Chawla, and Meng Jiang. 2021. Modeling co-evolution of attributed and structural information in graph sequence. *IEEE TKDE* (2021).
[30] Daheng Wang, Tong Zhao, Nitesh V Chawla, and Meng Jiang. 2021. Dynamic Attributed Graph Prediction with Conditional Normalizing Flows. In *ICDM*. IEEE, 1385–1390.
[31] Yiwei Wang, Wei Wang, Yuxuan Liang, Yujun Cai, and Bryan Hooi. 2020. Graphcrop: Subgraph cropping for graph classification. In *arXiv:2009.10564*.
[32] Yiwei Wang, Wei Wang, Yuxuan Liang, Yujun Cai, Juncheng Liu, and Bryan Hooi. 2020. Nodeaug: Semi-supervised node classification with data augmentation. In *KDD*. 207–217.
[33] Xingfei Wei, Zhi Wang, Zhiting Tian, and Tengfei Luo. 2021. Thermal Transport in Polymers: A Review. *Journal of Heat Transfer* 143, 7 (2021), 072101.
[34] Yingxin Wu, Xiang Wang, An Zhang, Xiangnan He, and Tat-Seng Chua. 2022. Discovering Invariant Rationales for Graph Neural Networks. In *ICLR*.
[35] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE TNNLS* 32, 1 (2020), 4–24.
[36] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. 2018. MoleculeNet: a benchmark for molecular machine learning. *Chemical science* (2018), 513–530.
[37] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *ICLR*.
[38] Jason Yang, Lei Tao, Jinlong He, Jeffrey McCutcheon, and Ying Li. 2021. Discovery of Innovative Polymers for Next-Generation Gas-Separation Membranes using Interpretable Machine Learning. In *chemrxiv-2021-p4g7z*.
[39] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. Gnnexplainer: Generating explanations for graph neural networks. In *NeurIPS*.
[40] Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L Hamilton, and Jure Leskovec. 2018. Hierarchical graph representation learning with differentiable pooling. In *NeurIPS*. 4805–4815.
[41] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. In *NeurIPS*. 5812–5823.
[42] Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2020. Deep learning on graphs: A survey. *IEEE TKDE* (2020).
[43] Tong Zhao, Tianwen Jiang, Neil Shah, and Meng Jiang. 2021. A synergistic approach for graph anomaly detection with pattern mining and feature learning. *IEEE TNNLS* (2021).
[44] Tong Zhao, Gang Liu, Stephan Günnemann, and Meng Jiang. 2022. Graph Data Augmentation for Graph Machine Learning: A Survey. *arXiv preprint arXiv:2202.08871* (2022).
[45] Tong Zhao, Gang Liu, Daheng Wang, Wenhao Yu, and Meng Jiang. 2022. Learning from Counterfactual Links for Link Prediction. *ICML* (2022).
[46] Tong Zhao, Yozen Liu, Leonardo Neves, Oliver Woodford, Meng Jiang, and Neil Shah. 2021. Data Augmentation for Graph Neural Networks. In *AAAI*. 11015.
[47] Tong Zhao, Bo Ni, Wenhao Yu, Zhichun Guo, Neil Shah, and Meng Jiang. 2021. Action Sequence Augmentation for Early Graph-based Anomaly Detection. In *CIKM*. 2668–2678.
[48] Jiajun Zhou, Jie Shen, and Qi Xuan. 2020. Data Augmentation for Graph Classification. In *CIKM*. 2341–2344.
[49] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2021. Graph contrastive learning with adaptive augmentation. In *WWW*. 2069–2080.

A DATASET DETAILS

Polymer datasets. The four datasets GlassTemp, MeltingTemp, PolyDensity, and O_2Perm are used to predict different properties of polymers such as *glass transition temperature* (°C), *polymer density* g/cm^3, *melting temperature* (°C), and *oxygen permeability* (Barrer). GlassTemp, MeltingTemp, and PolyDensity are collected from PolyInfo, which is the largest web-based polymer database [20]. The O_2Perm dataset is created from the Membrane Society of Australasia portal, consisting of a variety of gas permeability data [25]. However, the limited size (i.e., 595 polymers) brings great challenges to rationale identification and property prediction. Since a polymer is built from repeated monomer units, researchers use monomers as polymer graphs to predict properties. Different from molecular graphs, the monomer graphs have two special nodes (see "$*$" in the molecular structures in Figure 1), indicating the polymerization points of monomers [17]. For all the polymer datasets, we randomly split by 60%/10%/30% for training, validation, and test.

Molecule datasets. Besides polymer datasets, we use seven molecule datasets from the graph property prediction task on Open Graph Benchmark or known as OGBG. They were originally collected by MoleculeNet [36] and used to predict the properties of molecules, including (1) inhibition to HIV virus replication in ogbg-HIV, (2) toxicological properties of 617 types in ogbg-ToxCast, (3) toxicity measurements such as nuclear receptors and stress response in ogbg-Tox21, (4) blood–brain barrier permeability in ogbg-BBBP, (5) inhibition to human β-secretase 1 in ogbg-BACE, (6) FDA approval status or failed clinical trial in ogbg-ClinTox, and (7) having drug side effects of 27 system organ classes in ogbg-SIDER. For all molecule datasets, we use the scaffold splitting procedure as OGBG adopted [9]. It attempts to separate structurally different molecules into different subsets, which provides a more realistic estimate of model performance in experiments [36].

B IMPLEMENTATION DETAILS

All the experiments in this work are conducted on an Linux server with Intel Xeon Gold 6130 Processor (16 Cores @2.1Ghz), 96 GB of RAM, and a single RTX 2080Ti card (11 GB of RAM). Our method is implemented with `Python 3.9.9` and `PyTorch 1.10.1`. We manually tune the hyperparameters over the following ranges:

- $\gamma \in \{0.05, 0.1, 0.15, \ldots, 0.75, 0.8\}$,
- $T_{sep} \in \{1, 2\}$,
- $T_{pred} \in \{2, 3\}$,
- Learning rate $\in \{0.001, 0.005, 0.01\}$,
- Batch size $\in \{32, 128, 256, 512\}$,
- Representation dimensions $d_1, d_2 \in \{64, 128, 300\}$,
- Number of GNN_1 layer $L_1 = \{2\}$,
- Number of GNN_2 layers $L_2 \in \{2, 3, 4, 5\}$.

We use sum pooling as the default $\text{AGG}(\cdot)$ in GREA for the experiments in Tables 2 and 3. We set GIN as the default encoder for all ablation studies, case studies, and efficiency analysis. We employ the virtual node trick [9] for all methods on the ogbg-HIV, ogbg-Tox21, ogbg-BBBP, and all polymer datasets. For PolyDensity, we train and evaluate the models using the logarithm of the property [17]. We report the mean and standard deviation of the test performance over 10 runs with different random initialization of the parameters.

Our code and data are available on the GitHub[2]. To implement the baseline methods, we use the official code package[3] from the authors for DIR [34]. For U-NetsPool [6] and SelfAttnPool [14], we use the public implementation provided by the `PyG`[4] package. For IRM [1], we implement it's graph version based on its official repository.[5] As source codes of OOD-GNN [15] and StableGNN [5] are not publically available, we implement then with the official code package of StableNet[6] and the `PyG` package.

[2]https://github.com/liugangcode/GREA
[3]https://github.com/Wuyxin/DIR-GNN
[4]https://github.com/pyg-team/pytorch_geometric
[5]https://github.com/facebookresearch/InvariantRiskMinimization
[6]https://github.com/xxgege/StableNet