Rationalizing Graph Neural Networks with Data Augmentation

GANG LIU, ERIC INAE, TENGFEI LUO, and MENG JIANG, University of Notre Dame, USA

Graph rationales are representative subgraph structures that best explain and support the graph neural network (GNN) predictions. Graph rationalization involves the joint identification of these subgraphs during GNN training, resulting in improved interpretability and generalization. GNN is widely used for node-level tasks such as paper classification and graph-level tasks such as molecular property prediction. However, on both levels, little attention has been given to GNN rationalization and the lack of training examples makes it difficult to identify the optimal graph rationales. In this work, we address the problem by proposing a unified data augmentation framework with two novel operations on environment subgraphs to rationalize GNN prediction. We define the environment subgraph as the remaining subgraph after rationale identification and separation. The framework efficiently performs rationale–environment separation in the *representation space* for a node's neighborhood graph or a graph's complete structure to avoid the high complexity of explicit graph decoding and encoding. We conduct experiments on 17 datasets spanning node classification, graph classification, and graph regression. Results demonstrate that our framework is effective and efficient in rationalizing and enhancing GNNs for different levels of tasks on graphs.

CCS Concepts: • **Computing methodologies** → **Neural networks**; *Regularization*; • **Applied computing** → **Chemistry;**

Additional Key Words and Phrases: Graph neural network, node classification, graph property prediction, data augmentation, rationalization

ACM Reference Format:
Gang Liu, Eric Inae, Tengfei Luo, and Meng Jiang. 2024. Rationalizing Graph Neural Networks with Data Augmentation. *ACM Trans. Knowl. Discov. Data.* 18, 4, Article 86 (February 2024), 23 pages.
https://doi.org/10.1145/3638781

1 INTRODUCTION

Graphs broadly exist in computational social science and chemistry [11, 14, 19, 47]. In social and citation networks, nodes are the central focus, and the interactive patterns between them provide insight into important decision-making processes. In the fields of chemoinformatics and bioinformatics, molecules and polymers (macromolecules) are often represented as labeled graphs, where atoms serve as nodes and bonds as edges [11, 12, 24]. **Graph neural networks (GNN)** automate feature extraction from graph data through nonlinear functions and layers that aggregate

This work was supported by NSF IIS-2142827, IIS-2146761, IIS-2234058, CBET-2102592, and ONR N00014-22-1-2507.
Authors' address: G. Liu, E. Inae, T. Luo, and M. Jiang, University of Notre Dame, Notre Dame, Indiana, USA; e-mails: gliu7@nd.edu, einae@nd.edu, tluo@nd.edu, mjiang2@nd.edu.

ACM 1556-4681/2024/02-ART86
https://doi.org/10.1145/3638781

information from node neighborhoods. Node classification tasks leverage node representation for categorical label predictions. For graph classification and regression, graph pooling is added to different layers of the GNN architecture to reduce the graph size and preserve important structural information for final prediction [13, 17, 47]. As different fields involve different tasks on graphs, it is crucial to train, explain, and justify GNN predictions for specific problems at different levels.

Despite the advances of various GNNs, their predictions are not readily explainable in a manner that is accessible and understandable to humans and thereby cannot be justified in practice. Post-hoc methods [46] produce explanations using another model and may not fully capture the real relationships learned in GNN models. Selective rationalization is recently developed to pinpoint the rational features (rationales) that support a model's predictions *at the model training stage*. By selecting a small subset of input features and using them to support model predictions, decisions from black-box neural networks become transparent and interpretable to humans. While rationalization is extensively studied in text data [3], their applications to GNNs remain underexplored for both node classifications and graph property predictions.

Additionally, the limitations in terms of data size, such as the number of nodes and graphs, would lead to overfitting and poor generalizability for GNN predictions. Therefore, the rationalization may fail to capture robust and causal reasons for these predictions [3]. For instance, in semi-supervised node classification, there are often only 10 labeled nodes per class [14]. The lack of supervision in GNNs would result in their predictions relying on unreliable features that emerge from message passing [39] and also result in the failure of interpretability for the rationalization of node classification. Graph property prediction tasks suffer from similar issues. Popular benchmarks for molecular graph classification often have datasets that range from 1,000 to 10,000 graphs [42]. In polymer graph regression, the number of graphs in popular datasets is even smaller, typically around 600 [22]. In such scenarios, it is difficult to identify correct subgraphs that provide explanations and justification for graph property predictions.

In this work, we make the first attempt to rationalize GNN predictions for both node-level and graph-level tasks in a unified data augmentation framework, as presented in Figure 1. For node-level task, the prediction for node v_i relies on its surrounding neighborhood within graph structure g_i in graph neural networks [14, 43, 46]. We define the rationale subgraph of v_i, denoted by $g_i^{(r)}$, from a specific set of neighborhoods of v_i that could determine the label of v_i. The remaining neighbors of v_i are referred to as the environment neighbors or environment subgraph $g_i^{(e)}$. Note that nodes in the environment subgraph do not contribute to explaining or causing the label of v_i. For example, the specific topic of a paper node focusing on graph neural networks could be determined by references talking about graph machine learning and data mining, rather than all references, which may include loosely connected work intended for external reading of text mining. In graph-level tasks, the scaffold substructure [24] often serves as the rationale subgraph $g_i^{(r)}$. It plays a crucial role in determining the property y_i of a complete graph g_i. The remaining environment substructure $g_i^{(e)}$ is needed for other purposes such as chemical synthesis and validity [24]. In both tasks, the rationale subgraph $g_i^{(r)}$ potentially reveals the cause of the label y_i from the view of data generation. To achieve the correct identification of the rationale subgraphs, we use a separator function f_{sep}. However, training f_{sep} and f_{GNN} at the same time would easily fail due to the lack of training examples. Therefore, we propose two data augmentation operations based on the separated environment subgraphs $g_i^{(e)}$. These operations efficiently expand the limited training set by simulating the generation of g_i as the combination of the rationale subgraph with the environment that perturbs the graph structure *without changing the label*. Once f_{sep} separates rationales and environments, the rationales $g_i^{(r)}$ are first used to train the GNN as examples augmented by *environment removal*. Second, diverse environments from other graphs g_j within the

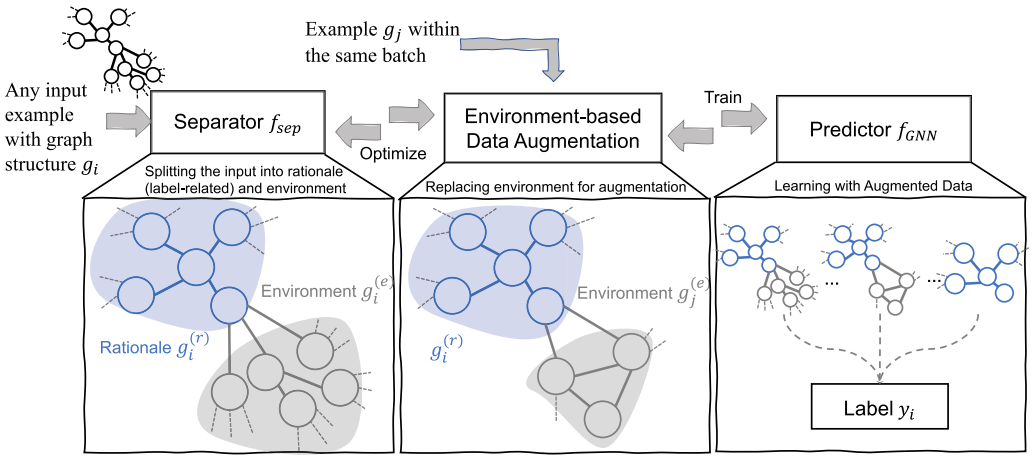


Fig. 1. An overview of RGDA. The input graph-structured data example could be a node in a large graph with surrounding neighbors or a graph in the graph dataset: (1) RGDA learns to identify rationale subgraphs with a rationale–environment separator f_{sep}; (2) different environment subgraphs from the same training batch are used to replace the original environment subgraph for data augmentation; and (3) the separator f_{sep} and GNN predictor f_{GNN} are jointly optimized using augmented training dataset. After training, f_{sep} supports explanations for f_{GNN} prediction with rationales.

training batch replace the original environment, thereby generating new examples: This augmentation is called *environment replacement*, i.e., $g_{(i,j)} = g_i^{(r)} \cup g_j^{(e)}$. f_{sep} is jointly optimized with f_{GNN} such that the identified rationales are robust to various prediction environments. Augmented examples are gradually improved with optimized rationales, resulting in better GNN explanations and predictions.

In this article, we propose a novel unified framework to **Rationalize GNN predictions with Data Augmentation (RGDA)** for both node-level and graph-level tasks based on environment subgraphs. We have two key challenges in the implementation. (1) *Efficiency*: The explicit rationale–environment separation and data augmentation in the structure space have high complexity in graph encoding and decoding. (2) *Effectiveness*: Since nodes are **not independent and identically distributed (Non-IID)**, removing or replacing the rationale neighborhood for a given node v_i can affect the decision-making process for another node v_j within a limited number of hops from node v_i. On the other side, splitting and combining molecular graphs are scientifically and technically difficult. Therefore, RGDA performs operations *in the representation space*. Specifically, a separator function is used to infer the probability of nodes being categorized into a rationale subgraph using a vector (with the opposite probability for the environment subgraph). For the task of node classification, RGDA employs the vector to identify rationale subgraphs during the process of message passing. Only rationale-specific messages are passed for the rationale subgraph representation. For graph property prediction tasks, the GNN is used to get the node representation and RGDA could identify the rationale and environment subgraphs in the node representation space. After obtaining the representation vectors for both the rationale and environment subgraphs in either task, we create new examples in which the environment is either replaced or removed with these representation vectors. This is achieved by combining the vectors from the rationale subgraph with various representations from the environment subgraphs. A straightforward combination operation is sum. Besides this, we also empirically study other combination choices such as the mean, max, and concatenation. We conduct experiments on 17 node-level and graph-level

datasets. Results demonstrate the advantages of RGDA to rationalize and improve GNN predictions over other baselines from robust and invariant learning. The main contributions of this work are summarized as follows:

- Novel interpretable GNN learning: RGDA is the first unified framework to rationalize and improve GNN with interpretable subgraphs for both node classification and graph property prediction;
- Novel graph data augmentation method: Two novel data augmentation operations are proposed to remove/replace the environment subgraphs separated from the nodes' neighborhood graphs or graphs' complete structures;
- Extensive experiments: On both nodes and graphs, the effectiveness and efficiency of RGDA are validated on 17 datasets. Node-level and graph-level explanations from rationales are also validated in case studies.

2 RELATED WORK

2.1 Graph Neural Networks

GNN such as **Graph Convolutional Network (GCN)** [14], **Graph Isomorphism Networks (GIN)** [43], Graph Attention Networks [34], and GraphSAGE [10] have been developed to automate representation learning with nonlinear functions from graph data for tasks such as node classification and graph property prediction [11, 35, 41]. However, the decision-making process of GNNs is hard to understand and not justified [15, 25]. In this work, we focus the extracting rationale subgraphs during GNN training to provide explanations and support GNN predictions.

Semi-suspervised Node Classification. GNNs effectively learn from Non-IID node data [14, 34, 45, 51] to address the problem of semi-supervised node classification. The information used by GNNs for each node decision is associated with a neighborhood graph [14, 43, 46], which is implicitly constructed during message passing [1, 29]. To improve GNN generalization, a recent study by Wu et al. [39] proposes a method for learning domain-invariant representations by editing the entire graph structure. However, this approach incurs significant computational cost and lacks interpretability.

Graph Property Prediction. GNNs are broadly used for predicting the categorical or numerical properties of molecules and polymers in drug and material discovery [11, 23, 27]. Since labeling of molecular graph examples typically requires specialized knowledge and expensive experiments [5, 33], the number of labeled training graphs is limited. Additionally, transparent decision-making in GNNs is crucial as it assists domain experts in accelerating the discovery of novel drugs and materials [26]. Recent efforts to model causality into GNN representation learning [7, 18] are not directly understandable to humans, and their performance may also be hindered by limited supervision.

Graph Rationalization. Selective rationalization identifies the small subset of input features by maximizing the predictive performance based only on the subset itself, called rationale. Although it has been extensively studied in natural language processing [2, 3, 30], graph rationalization remains underexplored. Very recently, a method called DIR has been proposed to identify rationale subgraphs for graph property prediction [40]. However, its performance is mainly evaluated on synthetic data and may not be optimal for identifying graph rationales in molecular property prediction, where the available number of training graphs is significantly limited. Additionally, DIR has high computational complexity and cannot rationalize GNN predictions on node-level tasks.

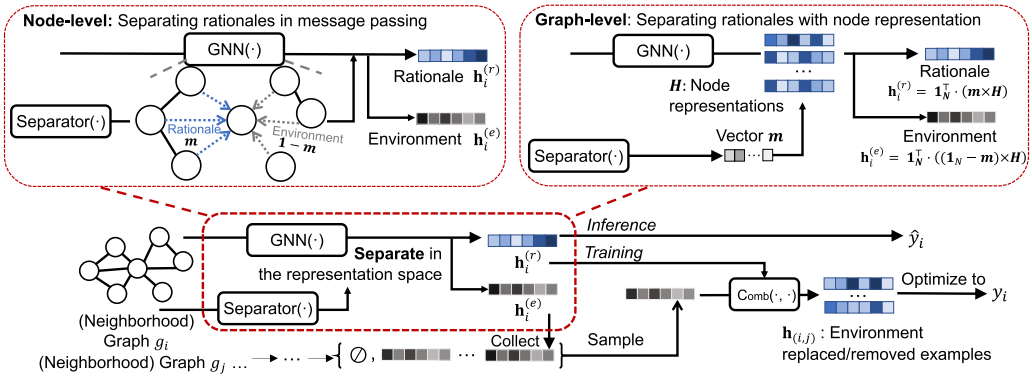


Fig. 2. The architecture of RGDA: It performs rationale–environment separation and data augmentation for node classification and graph property prediction in a unified framework. The representation learning of rationale/environment subgraphs and the creation of augmented examples with environment subgraphs are in *representation space*, instead of decoding every example into a graph form and running a GNN encoder on it. This design addresses the challenge of environment replacement for Non-IID node data and molecular graph data. It also aligns graph representation spaces and avoids high computational complexity.

2.2 Graph Data Augmentation

Graph data augmentation techniques [4, 49, 52] have improved the performance on semi-supervised node classification, such as DropEdge [29], NodeAug [38], and GAug [51]. Besides, many graph data augmentation 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 [36]. M-Evolve presented two heuristic algorithms including random mapping and motif-similarity mapping to generate weakly labeled data for small datasets [53]. MH-Aug adopted the Metropolis-Hastings algorithm to create augmented graphs from an explicit target distribution for semi-supervised learning [28]. SGIR [21] proposed the label-anchored mixup algorithm to create examples in the latent space and to address data imbalance issues. DCT [20] trained a diffusion model on unlabeled graph data and generated task-specific examples incorporating label information. This represents a novel approach to knowledge transfer through data augmentation. Meanwhile, graph contrastive learning learned unsupervised representations of graphs using graph data augmentations to incorporate various priors [48]. Zhu et al. [55] 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 graph data augmentation is given by Zhao et al.[49]. Our work proposes a unified data augmentation framework for node-/graph-level tasks based on environment subgraphs.

3 PROBLEM DEFINITION

Let $g = (\mathcal{V}, \mathcal{E})$ be a graph of $|\mathcal{V}|$ nodes and $|\mathcal{E}|$ edges, where $\mathcal{V}$ is the set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges. For each node $v \in \mathcal{V}$, there is a node feature is $\mathbf{x}_v \in \mathbb{R}^F$ with F dimensions. Given any node-level or graph-level task, GNN first learns the representation vector of the node $n \in \mathcal{V}$ in graph g by iteratively aggregating representations from v's neighbors $u \in \mathcal{N}(v)$. Formally, the

kth iteration (layer) of a GNN is

$$\begin{aligned} \mathbf{h}_{v,k} &= \mathrm{U}_k\left(\mathbf{h}_{v,k}, \mathbf{a}_{v,k}\right), \quad \text{where} \\ \mathbf{a}_{v,k} &= \mathrm{M}_k\left(\left\{\left(\mathbf{h}_{v,k-1}, \mathbf{h}_{u,k-1}\right), \forall u \in \mathcal{N}(v)\right\}\right). \end{aligned} \tag{1}$$

Here $\mathrm{U}_k(\cdot)$ and $\mathrm{M}_k(\cdot)$ are message and update functions at the kth layer. $\mathbf{h}_{v,k} \in \mathbb{R}^d$ is v's d-dimensional representation vector and the initial $\mathbf{h}_{v,0}$ is given by $\mathbf{x}_v$. The choice of U_k and M_k are task specific. A GNN-based node classifier/graph property predictor f_{GNN} stacks K layers of networks with Equation (1) to get the final node representation $\mathbf{h}_{v,K}$.

Node Classification. Given a single graph g, for each node $v \in \mathcal{V}$ on the graph, we have a categorical label $y_v \in \mathcal{Y}_{\mathcal{V}}$ associated with the node. With $\mathbf{h}_{v,K}$, GNN aggregates representation for the node v from a K-hop implicit ego-graph centered at v, as indicated by the neighborhood graphs in Figure 1 and Figure 2. f_{GNN} predicts the node label $\hat{y}_v$ by applying a softmax function to the last layer of node representation $\mathbf{h}_{v,K}$, which is denoted by $\mathbf{h}_v$ for brevity.

Graph Property Prediction. Suppose $g \in \mathcal{G}$ is the graph from the graph space $\mathcal{G}$. Graph property prediction pairs each graph with a label $y_g \in \mathcal{Y}_{\mathcal{G}}$, where $\mathcal{Y}_{\mathcal{G}}$ is the graph label space. It is categorical (classification) or numerical (regression). With GNN, f_{GNN} generates node representation matrix $\mathbf{H}$ as:

$$\mathbf{H} = \mathrm{GNN}(g) = \left[\ldots, \mathbf{h}_v, \ldots\right]^\top_{v \in \mathcal{V}} \in \mathbb{R}^{|\mathcal{V}| \times d}. \tag{2}$$

The graph representation $\mathbf{h}_g$ is summarized from $\mathbf{H}$ using a readout function (e.g., sum or mean), and f_{GNN} takes the final graph-level prediction $\hat{y}_g$ by a **Multilayer perceptron (MLP)**:

$$\mathbf{h}_g = \mathrm{readout}\left(\mathbf{H}\right) \in \mathbb{R}^d, \tag{3}$$

$$\hat{y}_g = \mathrm{MLP}(\mathbf{h}_g) \in \mathcal{Y}_{\mathcal{G}}. \tag{4}$$

Unfortunately, both node- and graph-level tasks suffer from lack of training examples. Besides, existing advances in GNN cannot justify the prediction from f_{GNN}. In the next section, we present a unified data augmentation framework to enhance GNN training under limited supervision and rationalize GNN predictions with explainable subgraphs.

4 PROPOSED FRAMEWORK

For brevity, we unify the notation of y_v and y_g as y and use y_i/y_j to indicate the label value of the node or graph with a index number i/j. In this section, we introduce the framework RGDA to rationalize and improve GNN predictions.

4.1 RGDA Overview

Figure 2 shows the overall training and inference architecture of RGDA. Although the implementation of the rationale–environment separation for node-level and graph-level tasks is slightly different, RGDA achieves the unification with the same philosophy that performs (1) rationale–environment separation with f_{sep} in Section 4.2 and (2) rationale–environment combination for data augmentation in the node representation space in Section 4.3. This unification comes from the fact that both node-level and graph-level tasks rely on a computation graph to extract the final node-level or graph-level representation for each prediction.

For node classification, the computation graph is implicitly built when we iteratively aggregate node representation from direct neighbors at each graph neural network layer. To predict the label of a node v, GNN aggregates information from a neighborhood graph g centered at v. In this case, we perform the rationale–environment separation in message passing. Specifically, we separate the passing of the rationale and environment messages to the center node v. It is achieved by

learning a vector $\mathbf{m}$ that infers whether the neighbor u is the rationale neighbor of the center node v. However, $1 - \mathbf{m}$ infers whether the neighbor u is the environment neighbor.

For graph property prediction, the computation graph is the input graph g itself. We first capture the contextualized representation of nodes with the K-layer GNN. Then we infer the rationale role of each node with the learned vector $\mathbf{m}$ and the environment role with $1 - \mathbf{m}$. Consequently, the representation of the rationale–environment subgraph is calculated by separately summarizing the representations of nodes that play different roles.

To this end, we get the representations for the rationale and environment and could simulate the rationale–environment combination for augmentation using a function $\text{Comb}(\cdot, \cdot)$. Augmented examples are created with gradually optimized rationale subgraphs, which also lead to better f_{GNN} predictions.

4.2 Rationale–Environment Separation

4.2.1 Separation for Node Classification. Given a node $v \in \mathcal{V}$ from the graph g, f_{sep} in RGDA separates v's neighbors into two roles: the rationale that supports and explains node prediction and the environment. A K-layer GNN implicitly constructs a K-hop neighborhood graph for node v_i, and the rationale–environment subgraph ($g^{(r)}/g^{(e)}$) of v can be similarly constructed by identifying the rationale and environment neighborhoods of v, which are presented in Figure 1. RGDA incorporates the rationale–environment separation in the message passing from Equation (1). Specifically, suppose $u \in \mathcal{V}$ is the v's direct neighbor and $m_{vu} = Pr_v(u \in \mathcal{N}(v))$ is the mask that indicates the probability of the neighbor $u \in \mathcal{N}(v)$ being classified into the rationale neighbor of v,

$$m_{vu} = \text{sigmoid}\left(\text{MLP}_{\text{sep}}([\mathbf{x}_v \| \mathbf{x}_u])\right), \tag{5}$$

where $[\cdot\|\cdot]$ is vector concatenation. In practice, we could build the mask vector $\mathbf{m} \in \mathbb{R}^{|\mathcal{E}|}$ for edges and use v and u to index each element in $\mathbf{m}$ as the m_{vu}. With m_{vu}, the rationale representation $\mathbf{h}^{(r)}$ of v's rationale subgraph $g^{(r)}$ is calculated from K layers of rational messages $\mathbf{a}^{(r)}$ in f_{GNN}. Specifically, at each layer, the rationale representation $\mathbf{h}^{(r)}_{v,k}$ is updated by the message from the rationale neighbors as

$$\begin{aligned} \mathbf{h}^{(r)}_{v,k} &= \text{U}_k\left(\mathbf{h}^{(r)}_{v,k}, \mathbf{a}^{(r)}_{v,k}\right), \quad \text{where} \\ \mathbf{a}^{(r)}_{v,k} &= \text{M}_k\left(\left\{\left(\mathbf{h}^{(r)}_{v,k-1}, \mathbf{h}^{(r)}_{u,k-1}, m_{vu}\right), \forall u \in \mathcal{N}(v)\right\}\right). \end{aligned} \tag{6}$$

Here $\mathbf{h}^{(r)}_{v,0}$ is initialized by $\mathbf{x}_v$ and m_{uv} is a scalar edge feature that weights the message from $\mathbf{h}^{(r)}_{u,k-1}$ [9]. For the environment representation $\mathbf{h}^{(e)}$ of v's environment subgraph $g^{(e)}$, we apply $1 - m_{uv}$ at the first layer of message passing,

$$\begin{aligned} \mathbf{h}^{(e)}_{v,1} &= \text{U}_1\left(\mathbf{x}_v, \mathbf{a}^{(e)}_{v,1}\right), \quad \text{where} \\ \mathbf{a}^{(e)}_{v,1} &= \text{M}_1\left(\{(\mathbf{x}_v, \mathbf{x}_u, 1 - m_{vu}), \forall u \in \mathcal{N}(v)\}\right). \end{aligned} \tag{7}$$

After that, the representation of the environment subgraph $\mathbf{h}^{(e)}$ is obtained by updating $\mathbf{h}^{(e)}_{v,1}$ with K-1 layers of standard message passing from Equation (1).

4.2.2 Separation for Graph Property Prediction. Given any graph $g \in \mathcal{G}$, f_{sep} in RGDA outputs a node-level mask vector $m_v = Pr(v \in \mathcal{V}^{(r)})$ to split g into rationale/environment subgraph. Specifically, m_v indicates the probability of node $v \in \mathcal{V}$ being classified into the rationale subgraph:

$$\mathbf{m} = \text{sigmoid}(\text{MLP}_{\text{sep}}(\text{GNN}_{\text{sep}}(g))). \tag{8}$$

GNN_{sep} in the separator extracts contextualized node representation for rationale–environment separation. Based on $\mathbf{m}$, we have $(\mathbf{1}_N - \mathbf{m})$, which indicates the probability of nodes being classified into the environment subgraph. Besides, GNN-based graph property predictor f_{GNN} generates node representation $\mathbf{H}$ with Equation (2): $\mathbf{H} = \text{GNN}(g)$. With $\mathbf{m}$ and $\mathbf{H}$, the rationale subgraph and environment subgraph can be easily separated in the latent space. Using sum pooling for readout, we have

$$\mathbf{h}^{(r)} = \mathbf{1}_N^\top \cdot (\mathbf{m} \times \mathbf{H}), \tag{9}$$

$$\mathbf{h}^{(e)} = \mathbf{1}_N^\top \cdot ((\mathbf{1}_N - \mathbf{m}) \times \mathbf{H}), \tag{10}$$

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.

4.3 Data Augmentations with Environment Subgraphs

Given B training graphs/nodes in the batch, the rationale–environment separator has generated the representations of rationale and environment subgraphs for each graph g_i or node v_i. That is, we have $\{(\mathbf{h}_1^{(r)}, \mathbf{h}_1^{(e)}), (\mathbf{h}_2^{(r)}, \mathbf{h}_2^{(e)}), \ldots, (\mathbf{h}_B^{(r)}, \mathbf{h}_B^{(e)})\}$. Given any two examples with indices $i, j \in \{1, 2, \ldots, B\}$, f_{GNN} without rationalization actually combines a rationale with a single environment to generate the (neighborhood) graph $g_i = \text{Comb}(g_i^{(r)}, g_i^{(e)})$ for node or graph prediction. Without rationalization, f_{GNN} may learn the spurious correlation between $g_i^{(e)}$ and y for accurate but not justified prediction. Without data augmentation, rationalization suffers from the small dataset and may fail to identify/learn a correct $g_i^{(r)}/\mathbf{h}_i^{(r)}$ to support accurate prediction. Therefore, RGDA creates training examples by simulating possible rationale–environment combinations in the node representation space using the combination function $\text{Comb}(\cdot, \cdot)$. For each rationale representation $\mathbf{h}_i^{(r)}$, it could be combined with any other environment representation $\mathbf{h}_j^{(e)}$ from the batch without changing the label value associated with y_i to directly create the representation $\mathbf{h}_{(i,j)}$ of a new example $g_{(i,j)}$ with label $y_{(i,j)} = y_i$.

4.3.1 Environment Removal Augmentation. Data augmentation with environment removal is a special case of $g_{(i,j)}$ by combing the rationale subgraph $g_i^{(r)}$ with an empty environment subgraph. Since the goal of RGDA is to rationalize GNN predictions by identifying the rationale subgraph, which is considered the causal factor behind the node label or graph property, the identified rationale itself should be good for GNN predictions. Specifically, given the rationale subgraph representation $\mathbf{h}_i^{(r)}$ of graph g_i, we apply the softmax function on $\mathbf{h}_i^{(r)}$ to take prediction $\hat{y}_i^{(r)}$ for the node v_i or modify Equation (4) as follows to predict $\hat{y}_i^{(r)}$ for the graph g_i:

$$\hat{y}_i^{(r)} = \text{MLP}\big(\mathbf{h}_i^{(r)}\big). \tag{11}$$

4.3.2 Environment Replacement Augmentation. Data augmentation with environment replacement replaces the environment subgraph of the (neighborhood) graph g_i with any other environment subgraph from a different example g_j. Given that the rationale subgraph $g_i^{(r)}$ is considered to determine the label of g_i, the environment subgraphs can be interpreted as natural noise on the rationale subgraphs. Hence, the replacement augmentation enhances the model's robustness against the noise signal brought by the environment subgraphs. For the graph property prediction task, for each graph g_i, we combine its rationale subgraph with all other $\tilde{B} = B - 1$ environment subgraphs $g_j^{(e)}, j \in \{1, 2, \ldots, B\} \setminus \{i\}$ in the batch. For node classification, since we often use all nodes in a single batch, we limit the number of environment subgraphs $\tilde{B}$ to a relatively small value and treat $\tilde{B}$ as a hyper-parameter. Therefore, the environment replacement data augmentation generates $\tilde{B}$

new (neighborhood) graphs, which would be optimized during training. As we directly get representations of the environment subgraphs, the combination function operates in the latent space as well to directly produce the representation of the $g_{(i,j)}$, e.g., $\mathbf{h}_{(i,j)} = \text{Comb}(g_i^{(r)}, g_j^{(e)})$. Specifically, the combination function can be any pooling functions like concatenation, sum pooling, and max pooling. Taking the element-wise sum pooling as an example, the representation $\mathbf{h}_{(i,j)}$ of a combined (neighborhood) graph of rationale subgraph $g_i^{(r)}$ and environment subgraph $g_j^{(e)}$ can be calculated as follows:

$$\mathbf{h}_{(i,j)} = \text{Comb}\left(g_i^{(r)}, g_j^{(e)}\right) = \mathbf{h}_i^{(r)} + \mathbf{h}_j^{(e)}. \tag{12}$$

Similarly to the removal augmentation, we predict the node label with $\mathbf{h}_{(i,j)}$ by applying a softmax function or predict the graph label with $\mathbf{h}_{(i,j)}$ by modifying Equation (4) as

$$\hat{y}_{(i,j)} = \text{MLP}\left(\mathbf{h}_{(i,j)}\right). \tag{13}$$

4.4 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 node/graph label. For example, when the node or graph label y is categorical value in the classification task, we use the standard cross-entropy loss. When the graph property y has real values in the graph regression task, we use the mean squared error loss. Without loss of generality, suppose we focus on the binary classification task. Given a batch of B training examples $g_1, g_2, \ldots, g_B$ (or $v_1, v_2, \ldots, v_B$), the loss functions for each example with the index i and its label y_i are defined as

$$\mathcal{L}_{rem} = y_i \cdot \log \hat{y}_i^{(r)} + (1 - y_i) \cdot \log\left(1 - \hat{y}_i^{(r)}\right), \tag{14}$$

$$\mathcal{L}_{rep} = \frac{1}{B} \sum_{j=1}^{B} \left(y_i \cdot \log \hat{y}_{(i,j)} + (1 - y_i) \cdot \log(1 - \hat{y}_{(i,j)})\right), \tag{15}$$

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:

$$\mathcal{L}_{reg} = \left| \frac{1}{N} \cdot \mathbf{1}_N^\top \cdot \mathbf{m} - \gamma \right|, \tag{16}$$

where $\gamma \in [0, 1]$ is a hyperparamter to control the expected size of the rationale subgraph $g^{(r)}$. $N = |\mathcal{V}|$ for graph property prediction and $N = \text{degree}(v)$ for node classification. Specifically, for graph-level tasks, we penalize the node number in the rationale when it deviates from our expectations. For node-level tasks, we penalize the number of rationale neighbors.

We use the alternate training schema in Chang et al. [3] to train RGDA. That is, we iteratively train f_{sep} and f_{GNN} for a fixed number of epochs T_{sep} and T_{pred}, respectively. The loss functions for training RGDA are

$$\mathcal{L}_{pred} = \mathcal{L}_{rem} + \alpha \cdot \mathcal{L}_{rep}, \tag{17}$$

$$\mathcal{L}_{sep} = \mathcal{L}_{rem} + \alpha \cdot \mathcal{L}_{rep} + \beta \cdot \mathcal{L}_{reg}, \tag{18}$$

where $\mathcal{L}_{pred}$ in Equation (17) and $\mathcal{L}_{sep}$ in Equation (18) are used to train f_{GNN} and f_{sep}, 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 node label of the input node v_i or the predicted property of input graph g_i. We present the algorithm for both model training and testing with RGDA in Alogrithm 1.

4.5 Complexity Analysis

We focus on the time complexity analysis of RGDA with GNNs. For each training batch, we assume there are n nodes and m edges. For graph-level tasks, we assume that there are n' graphs. We view them as a single graph with n nodes and disconnected components, which should not affect our complexity analysis for GNNs. We assume the graph is sparse and thereby we could conduct sparse matrix multiplication. For each layer of the GNN, the time complexity is $O(nd^2 + md)$ [50], where d is the dimension of the hidden representation vector. By stacking K layers of networks, the complexity of a GNN is $O(Knd^2 + Kmd)$. f_{sep} could be implemented with an individual GNN and thus has linear time complexity with respect to the standard GNN. The combination operation is conducted in the representation space, resulting in a linear complexity of n^2 for the node-level task and n'^2 for the graph-level task. Therefore, the overall time complexity of our method is $O(Knd^2 + Kmd + n^2)$ on the node tasks and $O(Knd^2 + Kmd + n'^2)$ on graph tasks.

5 EXPERIMENTS

We conduct comprehensive experiments for RGDA to answer the following research questions:

- *Effectiveness and Efficiency* in Section 5.2: In comparison to state-of-the-art methods that are specific to node classification, graph regression, and graph classification, can RGDA improve the performance of GNN?
- *Framework Analysis* in Section 5.3: What components contributes to the improvement from RGDA? Can the efficient and positive impact of data augmentation on environment subgraphs in the node representation space be credited for the improved model performance?
- *Interpretability* in Section 5.4: Are the rationale subgraphs identified by RGDA able to accurately and justifiably explain the GNN prediction?

5.1 Experimental Settings

5.1.1 Datasets. We conduct experiments on *six* node classification datasets, *four* polymer graph regression datasets, and *seven* molecular graph classification datasets. Statistical details of 17 datasets are in Table 1. We introduce them in details as follows.

Node Classification Datasets. Six datasets include three commonly used citation networks: Cora, CiteSeer, and PubMed from [14, 45], two amazon product networks: Photo and Computers, and a citation network: ogbn-Arxiv. Cora, CiteSeer, and Pubmed have 7, 6, and 3 classes respectively. Node features for these datasets are derived from the bag-of-words representation of scientific papers. The ogbn-Arxiv dataset, which has 40 classes, uses a 128-dimensional feature vector derived from the title and abstract of each node. The amazon product networks dataset consists of nodes representing Amazon goods, with edges indicating commonly co-purchased items. Node features for this dataset are obtained from the bag-of-words representation of product reviews. The Photo and Computers datasets have 8 and 10 classes respectively, indicating their product category. We perform transductive and semi-supervised learning for node classification and follow previous work to split the Cora, CiteSeer, PubMed, and ogbn-Arxiv [11, 14, 34]. For Photo and Computers datasets, we randomly label 10 nodes per class for training.

Graph Property Prediction Datasets. Four graph regression datasets are from polymer graphs. GlassTemp, MeltingTemp, PolyDensity, and O_2Perm are used to predict 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 [27]. The O_2Perm is from the Membrane Society of Australasia portal, consisting of a variety of gas permeability data [33]. However, the limited size (i.e., 595 polymers) brings great challenges

ALGORITHM 1: RGDA

Input: Task type, the prediction model $f_{sep}(\cdot)$, $f_{GNN}(\cdot)$, the input training/validation/testiang nodes or graphs $\{v_1, v_2, \ldots, v_B\}$ or $\{g_1, g_2, \ldots, g_B\}$ depending on the task type (graph-level or node-level), the number of training epoch *n_epochs*, the number of epochs to optimize the separator and the predictor (T_{sep} and T_{pred}) in an alternative way.

```
// During Model training
```

for epoch from 0 to *n_epochs* **do**
  Set $\text{FLAG}_{\text{optimizer}}$ to T_{sep} or T_{pred} according to the current epoch
  if the task type is the node level **then**
    Compute $\mathbf{m} = f_{sep}(v_i)$ with Equation (5)
    Compute $\mathbf{h}^{(r)}$ with $\mathbf{m}$ and Equation (12).
    Compute $\mathbf{h}^{(e)}$ with $1 - \mathbf{m}$, Equations (7) and (1)
  else
    `// For graph-level tasks`
    Compute $\mathbf{m} = f_{sep}(g_i)$ with Equation (8)
    Compute $\mathbf{h}^{(r)}$ with $\mathbf{m}$ and Equation (9)
    Compute $\mathbf{h}^{(e)}$ with $1 - \mathbf{m}$ and Equation (10)
  end if
  `// For clarity, we can operate on matrices to avoid the following loop`
  for $i = 0$ to B **do**
    `// Environment-removal augmentation`
    Compute $\hat{y}_i^{(r)}$ from $\mathbf{h}_i^{(r)}$ with Equation (11)
    `// Environment-replaced augmentation`
    for $j = 0$ to B and $j \neq i$ **do**
      Compute $\mathbf{h}_{(i,j)} = \text{Comb}(\mathbf{h}_i^{(r)}, \mathbf{h}_j^{(e)})$ with Equation (12).
      Compute $\hat{y}_{(i,j)}$ from $\mathbf{h}_{(i,j)}$ with Equation (13)
    end for
    if $\text{FLAG}_{optimizer}$ is T_{sep} **then**
      Compute $\mathcal{L}_{sep}$ for the example i with Equation (18).
    else
      Compute $\mathcal{L}_{pred}$ for the example i with Equation (17).
    end if
  end for
  Backpropagation with the averaged $\mathcal{L}_{\text{sep}}$ or $\mathcal{L}_{\text{pred}}$.
end for

```
// During model inference
```

for $i = 0$ to B in the validation or testing set **do**
  Compute $\hat{y}_i^{(r)} = f_{GNN}(g_i)$ or $f_{GNN}(v_i)$
end for

Output: $\hat{y}_1^{(r)}, \hat{y}_2^{(r)}, \ldots, \hat{y}_B^{(r)}$

to rationale identification and property prediction. Since a polymer is built from repeated units, researchers use a single unit as a graph to predict the property. For all the polymer datasets, we randomly split them by 60%/10%/30% for training, validation, and testing. *Seven graph classification datasets* are from molecular graphs in the **Open Graph Benchmark (OGBG)**. They were originally collected by MoleculeNet [42] 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

Table 1. Statistics of 17 Datasets for Node Classification, Graph Regression, and Graph Classification

Task Type	Dataset	# Graphs	# Nodes Per Graph (Avg./Max)	# Edges Per Graph (Avg./Max)	# Train	# Valid	# Test
Node Classification	Cora	1	2,708	5,278	140	500	1,000
	CiteSeer	1	3,327	4,552	120	500	1,000
	Photo	1	7,650	238,162	80	500	1,000
	Computers	1	13,752	491,722	100	500	1,000
	PubMed	1	19,717	44,338	60	500	1,000
	ogbn-Arxiv	1	169,343	1,166,243	90,941	29,799	48,603
Graph Regression	GlassTemp	7,174	36.7 / 166	79.3 / 362	4,303	718	2,153
	MeltingTemp	3,651	26.9 / 102	55.4 / 212	2,189	366	1,096
	PolyDensity	1,694	27.3 / 93	57.6 / 210	1,015	170	509
	O_2Perm	595	37.3 / 103	82.1 / 234	356	60	179
Graph Classification	ogbg-HIV	41,127	25.5 / 222	54.9 / 502	32,901	4,113	4,113
	ogbg-ToxCast	8,576	18.8 / 124	38.5 / 268	6,860	858	858
	ogbg-Tox21	7,831	18.6 / 132	38.6 / 290	6,264	783	784
	ogbg-BBBP	2,039	24.1 / 132	51.9 / 290	1,631	204	204
	ogbg-BACE	1,513	34.1 / 97	73.7 / 202	1,210	151	152
	ogbg-ClinTox	1,477	26.2 / 136	55.8 / 286	1,181	148	148
	ogbg-SIDER	1,427	33.6 / 492	70.7 / 1010	1,141	143	143

The Avgerage number and maximum number of nodes or edges per graph are also presented for graph-level tasks.

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 [11]. It attempts to separate structurally different molecules into different subsets, which provides a more realistic estimate of model performance in experiments [42]. Statistical details of the datasets are in Table 1.

5.1.2 Evaluation Metrics. We perform multi-class node classification and use the prediction accuracy as the metric. 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 [11, 22]. 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. We compare RGDA with different baselines specific for node-level and graph-level tasks.

Node Classification: We consider two basic GNN architectures and recent methods that improve generalization and data augmentation for GNNs. Specifically, they are as follows:

- GCN [14]: It introduces the first-order approximation of ChebNet [6] to construct the graph convolution layer for graph learning tasks.
- GraphSAGE [10]: It samples a fixed number of neighbors and employs batch training to learn the node representation on graphs.
- IRM [2]: The invariant risk minimization introduces a learning paradigm to estimate invariant predictors from multiple training environments. We adapt this paradigm for node classification tasks.
- SRGNN [54]: The Shift-Robust GNNs introduce a generalized framework to address the distributional shift problem in node classification tasks. This framework aids in training the GNNs, helping to avoid overfitting and improve generalization.
- EERM [39]: The explore-to-extrapolate risk minimization uses multiple graph structure editers as the context explorers to train the GNNs for better out-of-distribution generalization.

Table 2. Results on Node Classification: RGDA Consistently Achieves the Highest Accuracy (↑) %

		Cora	CiteSeer	Photo	Computers	PubMed	ogbn-Arxiv
GCN [14]	Standard	81.50 ± 0.62	71.28 ± 0.91	88.60 ± 1.70	81.13 ± 0.93	80.84 ± 0.21	71.60 ± 0.04
	IRM	81.87 ± 0.32	71.87 ± 0.31	88.82 ± 0.48	78.57 ± 0.87	77.63 ± 0.35	71.46 ± 0.34
	SRGNN	79.63 ± 1.22	69.90 ± 0.95	89.57 ± 0.38	80.50 ± 0.87	78.87 ± 0.75	68.00 ± 0.99
	EERM	81.03 ± 1.07	71.90 ± 0.10	89.55 ± 0.07	80.60 ± 1.98	79.30 ± 0.35	Out-of-Memory (>100G)
	MixUp	81.43 ± 0.38	71.63 ± 0.75	88.82 ± 0.74	80.08 ± 0.66	79.83 ± 0.59	71.55 ± 0.06
	EdgeDrop	81.67 ± 0.15	71.77 ± 0.50	88.30 ± 0.83	80.44 ± 0.78	80.40 ± 0.35	71.55 ± 0.16
	FLAG	81.90 ± 0.96	72.47 ± 0.31	89.54 ± 0.71	80.28 ± 0.19	80.20 ± 0.44	71.57 ± 0.24
	RGDA (Ours)	**84.33** ± 0.41	**73.02** ± 0.36	**89.76** ± 0.39	**83.32** ± 0.80	**82.08** ± 0.50	**72.88** ± 0.65
GraphSAGE [10]	Standard	80.64 ± 0.93	68.96 ± 0.68	85.93 ± 1.08	74.50 ± 1.22	78.52 ± 0.39	71.66 ± 0.14
	IRM	79.50 ± 1.13	70.10 ± 0.26	84.47 ± 2.72	72.37 ± 0.70	76.63 ± 1.02	70.95 ± 0.41
	SRGNN	76.67 ± 1.40	67.17 ± 0.51	86.13 ± 0.90	74.27 ± 1.42	75.53 ± 1.19	70.17 ± 0.43
	EERM	79.90 ± 0.87	69.83 ± 0.40	86.55 ± 0.21	73.25 ± 0.64	75.57 ± 2.15	Out-of-Memory (>100G)
	MixUp	80.77 ± 0.85	68.77 ± 0.64	85.88 ± 1.71	75.14 ± 0.95	77.53 ± 0.93	71.52 ± 0.15
	EdgeDrop	81.07 ± 0.85	70.30 ± 0.46	86.42 ± 1.54	75.00 ± 2.00	78.33 ± 0.76	71.47 ± 0.14
	FLAG	78.83 ± 0.15	70.20 ± 1.45	85.76 ± 1.75	75.20 ± 1.25	78.00 ± 0.40	71.59 ± 0.33
	RGDA (Ours)	**83.39** ± 0.69	**72.64** ± 0.67	**88.50** ± 0.85	**76.44** ± 1.33	**82.12** ± 0.57	**72.26** ± 0.24

Best Mean is **bold** and Best Baseline is underlined. The performance of MixUp [37] and EdgeDrop [29] on Cora, CiteSeer, and PubMed is different from the original report, since we follow the standard data splitting from References [14, 45] as presented in Table 1, in which much fewer training nodes are used.

- MixUp [37]: It adapts the MixUp algorithm to create more node examples for the node classification tasks. Specifically, the mixup operation is conducted for each layer of GNNs after message passing.
- EdgeDrop [29]: To improve the generalization of GNNs, it conducts data augmentation on graphs by randomly removing edges. A certain number of edges are removed at each training epoch to create different views of neighborhood structures for nodes.
- FLAG [16]: It introduces the free large-scale adversarial augmentation algorithm on graphs. The method perturbs node features with small adversarial noise, which is crafted by gradient ascent.

Graph Property Prediction: We consider two standard GNN architectures: GCN [14] and GIN [43] encoders. We also consider baselines that improve graph pooling, graph-level generalization, and graph-level rationalization techniques.

- GIN [43]: The graph isomorphism network uses MLP to update node representation during message passing. It is provably as powerful as the Weisfeiler–Lehman graph isomorphism test [32] for the graph-level tasks.
- U-NetsPool [8]: It introduces pooling and un-pooling operations on graphs, which encode the graph structures with a small set of nodes that can be recovered to the whole graphs.
- SelfAttnPool [17]: It uses the self-attention mechanism as the downsampling strategy on graph data to consider both node features and graph structure.
- StableGNN [7]: The Stable GNN formulates a general GNN learning framework to avoid spurious correlations during training. It uses differentiable graph pooling to extract subgraph representation and a regularizer to correct the biased training distribution with sample weights.
- OOD-GNN [18]: It trains the GNN for out-of-distribution graph-level tasks. It minimizes the statistical dependence between relevant and irrelevant graph representations through iterative optimization of the sample graph weights and graph encoder.

Table 3. Results on Graph (Polymer) Regression Datasets: RGDA 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 [14] as encoder	Standard	0.844 ± 0.007	44.1 ± 1.0	0.708 ± 0.040	60.9 ± 4.0	0.684 ± 0.028	0.090 ± 0.004	0.870 ± 0.147	688 ± 385
	U-NetsPool [8]	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 [17]	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 [7]	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 [18]	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 [2]	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 [40]	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
	RGDA−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
	RGDA (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 [43] as encoder	Standard	0.860 ± 0.006	41.8 ± 0.9	0.724 ± 0.008	59.4 ± 0.8	0.653 ± 0.024	0.095 ± 0.003	0.899 ± 0.075	658 ± 215
	U-NetsPool [8]	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 [17]	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 [7]	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 [18]	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 [2]	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 [40]	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
	RGDA−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
	RGDA (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

Best Mean is **bold** and best baseline is underlined.

- IRM [2]: We also consider the invariant risk minimization learning paradigm on graph-level tasks.
- DIR [40]: It is proposed by Wu et al. [40] to identify the subgraph that causes the graph-level labels during GNN training by intervening on the predicted logits.

Particularly, 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:

- DIR+RepAug: We add environment-replaced augmentation to DIR [40] to identify rationales, however, it has to explicitly decode and encode the rationales.
- RGDA−RepAug: We disable the environment replacement augmentation and use only the environment removal augmentation.

5.1.4 Implementation Details. All experiments are conducted on a Linux server with an 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 use sum pooling as the default Comb(·, ·) in RGDA for the experiments in Table 3 and Table 4. We set GCN and GIN as the default encoder for node-level and graph-level framework analysis, respectively. We utilize the virtual node trick [11] for all graph-level 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 [22]. We report the mean and standard deviation of the test performance over 10 runs with different random initialization of the parameters.

5.2 Effectiveness and Efficiency of RGDA

5.2.1 Improve GNN for Node Classification. From Table 2, RGDA outperforms all other baselines on six multi-class node classification tasks, achieving the highest accuracy on each dataset. We have the following observations:

Table 4. Results on Graph (Molecule) Classification Datasets: RGDA Consistently Achieves the Highest AUC (↑)

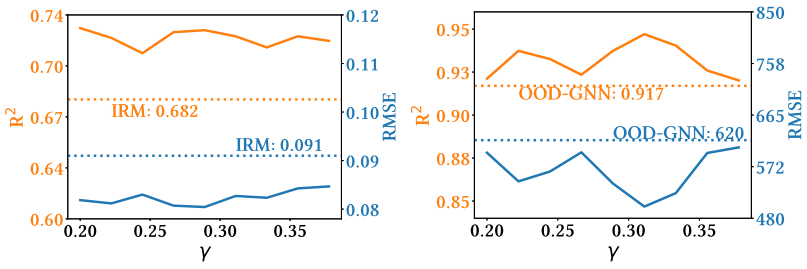
		ogbg-HIV	ogbg-ToxCast	ogbg-Tox21	ogbg-BBBP	ogbg-BACE	ogbg-ClinTox	ogbg-SIDER
GCN [14] as encoder	Standard	75.99 ± 1.19	65.80 ± 0.54	77.27 ± 0.82	67.77 ± 0.93	81.21 ± 2.47	88.47 ± 1.98	60.63 ± 1.01
	U-NetsPool	75.27 ± 1.04	65.07 ± 0.86	74.92 ± 0.93	67.09 ± 1.76	77.57 ± 1.73	84.50 ± 4.03	61.81 ± 1.21
	SelfAttnPool	77.33 ± 1.87	65.10 ± 0.76	75.63 ± 0.80	66.02 ± 2.20	73.83 ± 5.41	82.91 ± 7.91	57.18 ± 2.19
	StableGNN	72.18 ± 0.99	65.20 ± 1.09	74.54 ± 0.59	65.52 ± 1.84	66.07 ± 5.00	76.81 ± 7.78	56.44 ± 2.74
	OOD-GNN	75.80 ± 1.76	66.13 ± 0.46	76.73 ± 1.09	67.95 ± 1.65	80.96 ± 1.32	88.74 ± 1.43	61.33 ± 0.95
	IRM	77.02 ± 1.07	65.99 ± 0.63	76.54 ± 0.72	68.92 ± 0.53	79.47 ± 1.86	88.19 ± 2.31	60.35 ± 1.95
	DIR	74.66 ± 0.93	59.54 ± 1.54	47.27 ± 1.29	65.59 ± 2.98	67.51 ± 3.23	62.51 ± 9.56	53.31 ± 2.16
	DIR+RepAug	74.94 ± 2.25	66.32 ± 0.98	74.37 ± 0.54	66.30 ± 1.18	76.77 ± 2.26	86.06 ± 1.44	59.34 ± 1.70
	RGDA–RepAug	73.77 ± 2.10	66.14 ± 0.48	78.08 ± 0.61	67.36 ± 0.77	76.55 ± 5.29	87.08 ± 5.14	62.22 ± 1.66
	RGDA (ours)	**77.94** ± 0.65	**66.62** ± 0.41	**78.22** ± 0.93	**69.86** ± 1.75	**81.91** ± 2.40	**89.61** ± 1.50	**63.16** ± 1.51
GIN [43] as encoder	Standard	77.07 ± 1.49	66.59 ± 0.63	76.69 ± 0.64	68.08 ± 1.42	79.98 ± 2.01	86.82 ± 2.29	58.93 ± 2.41
	U-NetsPool	73.75 ± 3.62	65.24 ± 1.26	75.60 ± 0.93	68.09 ± 1.63	80.26 ± 1.05	81.46 ± 7.03	59.29 ± 1.14
	SelfAttnPool	75.33 ± 2.47	63.51 ± 1.37	75.07 ± 1.10	66.24 ± 1.67	73.48 ± 1.94	79.12 ± 9.95	57.02 ± 1.37
	StableGNN	72.18 ± 0.78	64.85 ± 0.25	73.81 ± 1.23	66.95 ± 1.20	72.29 ± 1.22	85.59 ± 2.24	55.93 ± 1.72
	OOD-GNN	77.99 ± 0.78	66.97 ± 0.51	76.46 ± 0.38	67.10 ± 1.88	78.00 ± 2.28	84.16 ± 4.96	59.16 ± 1.69
	IRM	78.17 ± 1.20	66.41 ± 0.65	75.42 ± 0.84	68.35 ± 0.71	79.77 ± 2.08	84.85 ± 2.15	57.78 ± 2.06
	DIR	75.33 ± 1.17	59.27 ± 0.97	50.78 ± 3.13	58.43 ± 4.43	61.15 ± 5.87	69.11 ± 8.10	54.06 ± 1.27
	DIR+RepAug	77.25 ± 2.49	64.54 ± 0.61	74.53 ± 0.80	68.13 ± 2.03	75.90 ± 6.42	85.61 ± 1.59	57.30 ± 1.15
	RGDA–RepAug	77.70 ± 1.78	66.81 ± 0.66	76.90 ± 1.17	67.37 ± 2.35	79.97 ± 3.80	85.74 ± 4.42	59.88 ± 1.69
	RGDA (ours)	**79.32** ± 0.92	**67.50** ± 0.67	**77.23** ± 1.19	**69.70** ± 1.28	**82.37** ± 2.37	**87.89** ± 3.68	**60.14** ± 2.04

Best Mean is **bold** and best baseline is underlined.

- *RGDA improves the prediction accuracy of GNN across different architectures and tasks*: Baseline methods may improve GNN performance when the graph is relatively small, but may not consistently achieve improvements across architectures and tasks. While FLAG [16] improves standard GCN on datasets with less than 10,000 nodes, such as Cora, CiteSeer, and Photo, it fails to do so on large-scale datasets such as Computers, PubMed, and ogbn-Arxiv. Although Mixup improves GCN performance on Photo, it underperforms when applied to GraphSAGE. On larger-scale datasets such as Computers, PubMeds, and ogbn-Arxiv, baseline methods consistently underperform standard GCN and GraphSAGE architectures. However, even on the largest dataset ogbn-Arxiv, RGDA still achieves improvements of +1.8% over GCN.
- *By simulating rationale–environment combinations with environment subgraphs, RGDA efficiently and effectively improves GNN generalization*: In comparison to standard GCN and GraphSAGE, node-level generalization methods such as SRGNN [54] and EERM [39] show minimal improvement due to the lack of labeled data. Besides, EERM [39] explicitly maintains different prediction environments through graph structure editing, this approach is computationally inefficient and not scalable to large graphs such as ogbn-Arxiv, which has over a million nodes. By collecting identified environment subgraphs in the representation space, RGDA efficiently performs data augmentation in the latent space and create many virtual training nodes to enhance GNN optimization.

5.2.2 Improve GNN for Graph Property Prediction. Table 3 presents the results on polymer property regression with R^2 and RMSE metrics. Table 4 presents the results on molecule property classification using AUC. Our RGDA achieves the best performance on all graph-level tasks with observations:

- *On both graph regression and classification tasks, RGDA demonstrates stable improvement*: While OOD-GNN is competitive with RGDA on regression tasks due to its elimination of



(a) PolyDensity (b) O_2Perm

Fig. 3. On two polymer datasets, the performance of RGDA is *not* sensitive to rationale size γ with wide ranges for tuning.

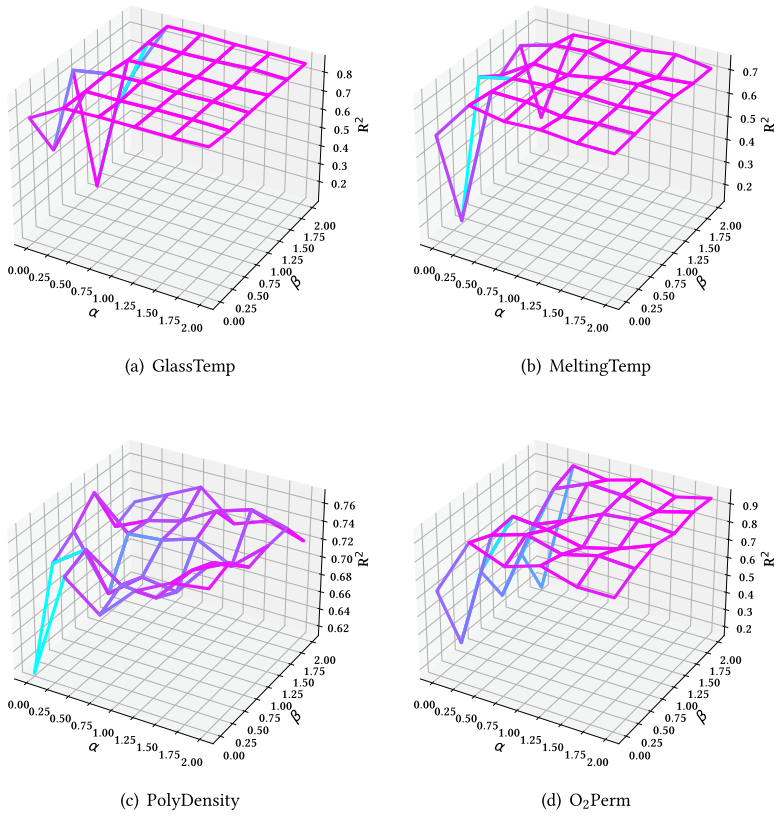
the statistical dependence between property-relevant and property-irrelevant graph representations, it underperforms other baselines on graph classification tasks. In contrast, RGDA achieves the best performance for both regression and classification tasks at the graph level, because it further utilizes property-irrelevant environment subgraphs to enhance rationale identification and enrich the limited labeled training data. So on datasets such as ogbg-ClinTox, RGDA with GIN improves AUC over OOD-GNN relatively by +4.4%.

- *RGDA effectively uses contextual information of atom (nodes) and data augmentation to improve graph-level rationalization*: The existing graph-level rationalization method DIR was evaluated on synthetic data [40] and it performs poorly on real polymer and molecule datasets. One reason is that the intervention in DIR is designed for classification but not suitable for regression tasks. Besides, DIR creates rationale subgraphs in the input space and re-encodes the subgraph with in-complete contextual information and half-trained GNN encoder. Besides, DIR suffers from a lack of labeled data to identify correct rationales. Compared to it, our RGDA achieves the best by performing rationale–environment separation in the contextual representation space and combining rationales with diverse environments to enhance rationale identification. So on datasets such as MeltingTemp, RGDA, with GIN produces 1.56×R^2 over DIR.

5.3 Framework Analysis of RGDA

5.3.1 Ablation Study on Data Augmentation. Tables 3 and 4 have presented the results of ablation studies of DIR+RepAug and RGDA−RepAug. DIR+RepAug is a variant of baseline method DIR by enabling *environment replacement augmentations* for training. RGDA−RepAug is a variant of our RGDA 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 RGDA−RepAug is not satisfactory. Environment replacement augmentations are effective for training graph rationalization methods.

5.3.2 Sensitivity Analysis. Without losing the generality, we conduct three series of sensitivity analyses. First, Figure 4 shows that on four polymer datasets, the performance of RGDA in terms of R^2 is insensitive to the hyperparameters α and β in Equation (18). Second, Figure 3 shows that the performance is insensitive to rationale size γ in Equation (16). Third, we compare the effects of different choices of Comb($\cdot$) function that aggregates the representations of rationale and environment subgraphs. Results are in Tables 5 and 6. We find that in most cases, we could define Comb($\cdot$) as the sum operation. Therefore, we can effectively and efficiently aggregate the rationale and the environment subgraphs in the latent space by adding their representation vectors together.



(a) GlassTemp (b) MeltingTemp

(c) PolyDensity (d) O_2Perm

Fig. 4. On four polymer datasets, the performance of RGDA (in R^2) is *not* sensitive to hyperparameters α and β in Equation (18).

Table 5. Investigation of the Combination Choice: Comb(·) in RGDA on Molecule Datasets

Comb(·) in RGDA	ogbg-HIV	ogbg-ToxCast	ogbg-Tox21	ogbg-BBBP	ogbg-BACE	ogbg-ClinTox	ogbg-SIDER
GCN [14] as encoder							
Sum Operation	77.94 ± 0.65	66.62 ± 0.41	**78.22** ± 0.93	**69.86** ± 1.75	**81.91** ± 2.40	**89.61** ± 1.50	**63.16** ± 1.51
Mean Operation	78.26 ± 1.26	64.95 ± 0.68	77.07 ± 0.40	69.41 ± 0.78	74.05 ± 6.18	89.12 ± 0.79	56.92 ± 0.80
Max Operation	77.88 ± 1.07	**67.42** ± 0.82	77.42 ± 0.88	66.48 ± 1.58	80.07 ± 2.98	84.66 ± 4.39	59.78 ± 2.06
Concatenation	**78.72** ± 0.76	67.23 ± 0.53	77.07 ± 0.65	67.31 ± 1.78	81.78 ± 1.83	87.98 ± 2.39	59.61 ± 2.38
GIN [43] as encoder							
Sum Operation	**79.32** ± 0.92	**67.50** ± 0.67	**77.23** ± 1.19	**69.70** ± 1.28	**82.37** ± 2.37	87.89 ± 3.68	60.14 ± 2.04
Mean Operation	78.10 ± 1.17	64.66 ± 0.69	76.48 ± 0.85	69.38 ± 1.95	76.55 ± 4.94	84.47 ± 4.78	58.45 ± 0.76
Max Operation	78.09 ± 1.37	67.01 ± 0.70	77.25 ± 0.38	68.23 ± 1.76	81.06 ± 1.84	86.72 ± 1.40	59.29 ± 2.01
Concatenation	77.71 ± 0.96	66.93 ± 1.23	77.29 ± 1.15	67.09 ± 1.87	80.84 ± 2.17	**88.89** ± 4.01	**60.78 ± 1.87**

We could choose Sum Operation by default, because it performs best in most molecular classification tasks.

5.3.3 Node-level Analysis for Effectiveness and Efficiency. We conduct experiments with GCN [14] to compare the effectiveness and efficiency of the separation and data augmentation performed in the graph structure space and the node representation space. We combine the rationale with one additional environment subgraph. We measure the test accuracy as the effectiveness metric and

Table 6. Investigation of the Combination Choice: Comb(·) in RGDA on Polymer Datasets

Comb(·) in RGDA	GlassTemp		MeltingTemp		PolyDensity		O_2Perm	
	R^2 ↑	RMSE ↓	R^2 ↑	RMSE ↓	R^2 ↑	RMSE ↓	R^2 ↑	RMSE ↓
GCN [14] as encoder								
Sum Operation	0.855 ± 0.003	42.6 ± 0.5	0.716 ± 0.015	60.2 ± 1.6	0.717 ± 0.023	0.086 ± 0.004	**0.941** ± 0.018	523.6 ± 91.0
Mean Operation	**0.858** ± 0.005	**42.1** ± 0.7	0.707 ± 0.021	61.1 ± 2.3	**0.739** ± 0.014	**0.082** ± 0.002	0.881 ± 0.113	684.5 ± 311.7
Max Operation	0.855 ± 0.005	42.5 ± 0.7	0.705 ± 0.016	61.4 ± 1.7	0.722 ± 0.014	0.085 ± 0.002	0.939 ± 0.050	**502.2** ± 194.6
Concatenation	0.854 ± 0.006	42.7 ± 0.8	**0.727** ± 0.012	**59.1** ± 1.3	0.732 ± 0.004	0.083 ± 0.001	0.927 ± 0.029	577.8 ± 119.9
GIN [43] as encoder								
Sum Operation	0.864 ± 0.005	41.2 ± 0.8	**0.736** ± 0.012	**58.0** ± 1.3	0.724 ± 0.030	0.085 ± 0.00	**0.930** ± 0.020	569.4 ± 85.6
Mean Operation	**0.868** ± 0.006	**40.6** ± 0.9	0.733 ± 0.007	58.4 ± 0.7	0.755 ± 0.011	0.080 ± 0.002	0.929 ± 0.033	**569.2** ± 121.5
Max Operation	0.858 ± 0.006	42.2 ± 0.9	0.7164 ± 0.009	60.2 ± 1.0	0.740 ± 0.012	0.082 ± 0.002	0.898 ± 0.049	676.6 ± 161.0
Concatenation	0.858 ± 0.008	42.1 ± 1.2	0.715 ± 0.013	60.4 ± 1.3	**0.757** ± 0.025	**0.079** ± 0.004	0.924 ± 0.014	598.6 ± 59.2

training time (in seconds) as the efficiency metric from 10 runs. Results in Table 7 demonstrate that performing rationale–environment separation and data augmentation in the representation space leads to a significant improvement in performance compared to operating in the graph structure space. Given that node data is Non-IID, it is challenging to perform precise replacement of environment neighborhoods for data augmentation. Besides, operations in the representation space reduce the training time.

5.3.4 Graph-level Analysis for Rationalization Efficiency. We first conduct analysis using the ogbg-HIV dataset to compare the rationalization efficiency of our method with DIR. Results are presented in Figure 5. When batch size increases, in other words, when a batch has more and more graphs, the time cost per batch of DIR increases significantly; and RGDA spends much less time than DIR. Empirically we show that our RGDA is more efficient than DIR. This is because RGDA does not explicitly decode or encode the subgraphs but directly creates their representations in latent space. Figure 5(b) shows that compared to three most competitive baselines, RGDA delivers the highest AUC by learning augmented examples, while spending comparable amount of training time.

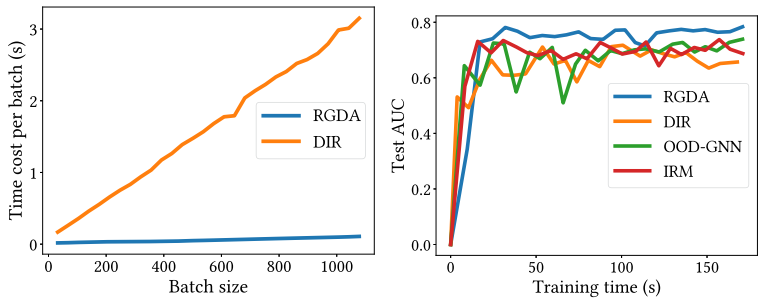
5.4 Case Study for Interpretability

5.4.1 Node-level Interpretability. In Figure 6, we present four cases where we use RGDA to rationalize GNN predictions for the Cora and PubMed datasets. We visualize the rationale subgraphs (in blue) to explain and understand why the decision to the nodes (in red) is right or wrong. Nodes with the same shape (triangle or circle) have the same label. From Figure 6, both GCN and GraphSAGE rely on neighbor nodes with the same label to make correct predictions. When the neighborhood is full of different shape nodes with different labels, node predictions from GNN may be wrong. However, in some cases, such as Figure 6(e), when the center circle node is connected to a circle node and a triangle node (that has a different label), RGDA may help identify the rationale subgraph in which most nodes share the same label and remove the message passing pathways from the environment subgraphs.

5.4.2 Graph-level Interpretability. Given test polymer examples in the O_2Perm dataset, we visualize and compare the rationale subgraphs that are identified by from DIR [40] and our RGDA in Figure 7. We have three observations. *First, the rationales identified by RGDA 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 RGDA estimates the probability of *nodes* being included in the rationales and uses the *contextualized representations* of atoms in the input graphs to create the representations of

Table 7. Node-level Effectiveness (Acc.) and Efficiency (Time) Comparison of Separation/Augmentation Performed in Representation (rep.) or Structure (struc.) Space

Space	Metric	Cora	CiteSeer	PubMed
Rep.	Acc.	83.95 ± 0.67	72.24 ± 0.95	81.60 ± 0.72
	Time (s)	2.22 ± 0.30	3.23 ± 0.38	3.94 ± 0.44
Struc.	Acc.	66.56 ± 0.88	62.82 ± 1.43	73.24 ± 2.39
	Time (s)	2.93 ± 0.45	3.84 ± 0.51	4.44 ± 0.45



(a) RGDA runs faster than DIR when batch size (# graphs) increases.

(b) RGDA spends comparable training time to deliver the highest AUC.

Fig. 5. Graph-level efficiency analysis on the ogbg-HIV dataset.

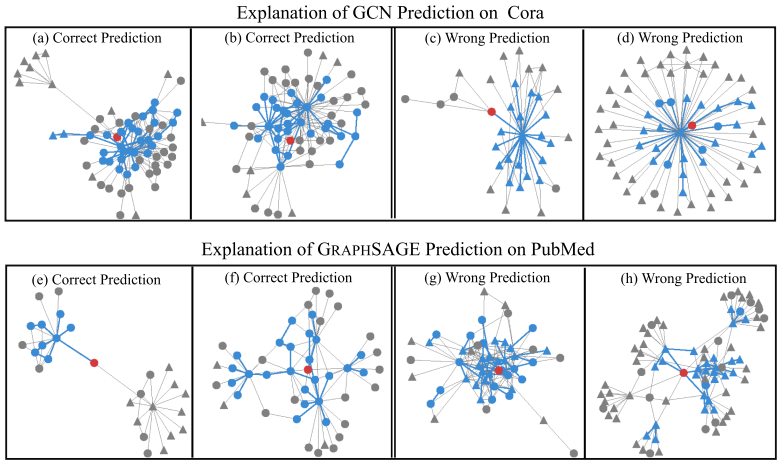


Fig. 6. Node-level interpretability: We visualize eight rationale subgraphs (neighborhoods) that justify the correct or wrong GNN node decisions for the center red nodes. The rationale subgraphs (including nodes and edges) are colored in blue, and the remaining subgraphs are colored in gray, representing the environments. Nodes with the same label are shown in the same shape.

rationales. So the rationales have coherent structures of nodes. *Second, the rationales from RGDA 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 7. The rationale from RGDA 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

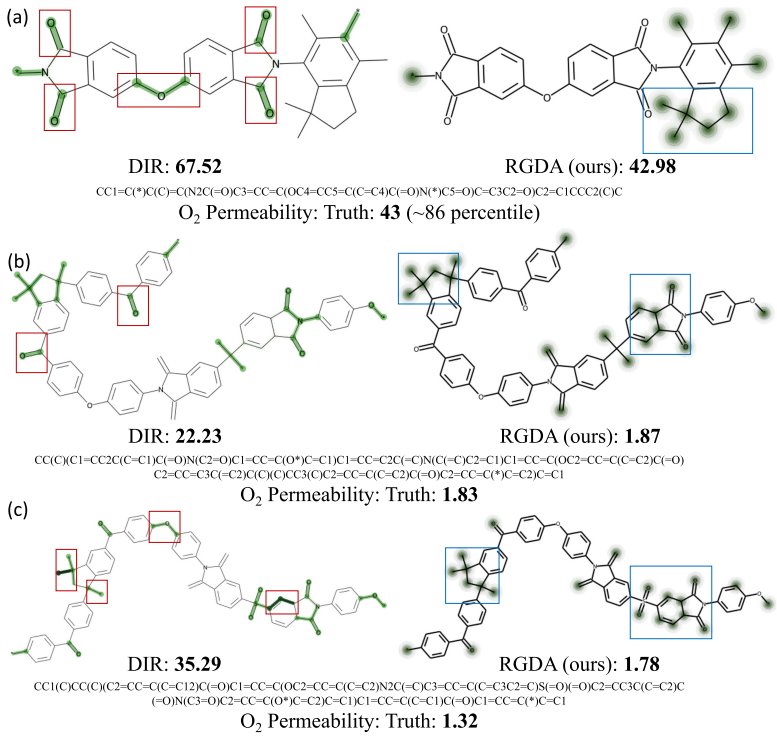


Fig. 7. Graph-level interpretability: three polymers selected from the test set in O_2Perm dataset to compare identified graph rationales by DIR [40] and our RGDA. DIR selects *edges* to decode rationale subgraphs. Our RGDA estimates the probability of *nodes* being classified into rationales in latent 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.

contributes to the gas permeability [31, 44]. The latter group is hydrophobic and contributes to steric frustration between polymer chains [44], inducing a positive correlation to the permeability. However, the rationale from DIR would make property predictor overestimate the oxygen permeability, because it suggests that the double-bonded oxygens, ethers, and nitrogen atoms are positively correlated with the property. However, it conflicts with observations and conclusions from chemical experiments in previous literature [44] 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 RGDA 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. RGDA intrinsically discovers correct relationships between rationale subgraphs and the property. *Third, the rationales from RGDA are commonly observed across different polymers.* We expect rationales to have a universal indication on the polymer properties. The rationales identified in the second and third examples both have fused heterocyclic rings (at the right end of the monomers and highlighted by blue boxes).

6 CONCLUSIONS

Despite the advances made by GNN, their predictions remain difficult to justify due to the lack of intrinsic interpretability. Our work addressed this problem with a novel data augmentation

framework, called RGDA. It is a unified solution to train robust, accurate, and interpretable GNN for both node-level and graph-level tasks with environment subgraph based augmented examples. We conduct experiments on 17 datasets using three commonly used GNN models for node classification, graph classification, and graph regression tasks. Our results demonstrate the effectiveness, efficiency, and interpretability of the framework.

REFERENCES

[1] Sami Abu-El-Haija, Amol Kapoor, Bryan Perozzi, and Joonseok Lee. 2020. N-gcn: Multi-scale graph convolution for semi-supervised node classification. In *Uncertainty in Artificial Intelligence*. PMLR, 841–851.

[2] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. 2019. Invariant risk minimization. arXiv:1907.02893. Retrieved from https://arxiv.org/abs/1907.02893

[3] Shiyu Chang, Yang Zhang, Mo Yu, and Tommi Jaakkola. 2020. Invariant rationalization. In *International Conference on Machine Learning*. PMLR, 1448–1458.

[4] 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 *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 3438–3445.

[5] Peter A. G. Cormack and Amaia Zurutuza Elorza. 2004. Molecularly imprinted polymers: Synthesis and characterisation. *J. Chromatogr. B* 804, 1 (2004), 173–182.

[6] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *Adv. Neural Inf. Process. Syst.* 29 (2016).

[7] Shaohua Fan, Xiao Wang, Chuan Shi, Peng Cui, and Bai Wang. 2024. Generalizing graph neural networks on out-of-distribution graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46, 1 (2024), 322–337.

[8] Hongyang Gao and Shuiwang Ji. 2021. Graph U-nets. *IEEE Trans. Pattern Anal. Mach. Intell.* (2021).

[9] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*. PMLR, 1263–1272.

[10] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 1025–1035.

[11] 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 *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS'20)*.

[12] Eric Inae, Gang Liu, and Meng Jiang. 2023. Motif-aware attribute masking for molecular graph pre-training. arXiv:2309.04589. Retrieved from https://arxiv.org/abs/2309.04589

[13] Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji, and Jiawei Han. 2023. Large language models on graphs: A comprehensive survey. arXiv:2312.02783. Retrieved from https://arxiv.org/abs/2312.02783

[14] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

[15] Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*. PMLR, 1885–1894.

[16] Kezhi Kong, Guohao Li, Mucong Ding, Zuxuan Wu, Chen Zhu, Bernard Ghanem, Gavin Taylor, and Tom Goldstein. 2022. Robust optimization as data augmentation for large-scale graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 60–69.

[17] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. 2019. Self-attention graph pooling. In *International Conference on Machine Learning*. PMLR, 3734–3743.

[18] Haoyang Li, Xin Wang, Ziwei Zhang, and Wenwu Zhu. 2023. OOD-GNN: Out-of-distribution generalized graph neural network. *IEEE Transactions on Knowledge and Data Engineering* 35, 7 (2023), 7328–7340.

[19] Gang Liu, Yong Deng, and Kang Hao Cheong. 2022. Network immunization strategy by eliminating fringe nodes: A percolation perspective. *IEEE Trans. Syst. Man Cybernet.: Syst.* 53, 3 (2022), 1862–1871.

[20] Gang Liu, Eric Inae, Tong Zhao, Jiaxin Xu, Tengfei Luo, and Meng Jiang. 2023. Data-centric learning from unlabeled graphs with diffusion model. *Adv. Neural Inf. Process. Syst.* (2023).

[21] Gang Liu, Tong Zhao, Eric Inae, Tengfei Luo, and Meng Jiang. 2023. Semi-supervised graph imbalanced regression. In *Proceedings of the 29th SIGKDD Conference on Knowledge Discovery and Data Mining*.

[22] Ruimin Ma and Tengfei Luo. 2020. PI1M: A benchmark database for polymer informatics. *J. Chem. Inf. Model.* 60, 10 (2020), 4684–4690.

[23] Ruimin Ma, Hanfeng Zhang, Jiaxin Xu, Luning Sun, Yoshihiro Hayashi, Ryo Yoshida, Junichiro Shiomi, Jian xun Wang, and Tengfei Luo. 2022. Machine learning-assisted exploration of thermally conductive polymers based on high-throughput molecular dynamics simulations. *Materials Today Physics* 28 (2022), 100850.

[24] Nidhi Mishra, Vinod K. Tiwari, and Richard R. Schmidt. 2020. Recent trends and challenges on carbohydrate-based molecular scaffolding: General consideration toward impact of carbohydrates in drug discovery and development. *Carbohydr. Drug Discov. Dev.* (2020), 1–69.

[25] Jiaming Mu, Binghui Wang, Qi Li, Kun Sun, Mingwei Xu, and Zhuotao Liu. 2021. A hard label black-box adversarial attack against graph neural networks. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (Virtual Event, Republic of Korea) (CCS'21)*. Association for Computing Machinery, New York, NY, USA, 108–125.

[26] Noushin Omidvar, Hemanth S. Pillai, Shih-Han Wang, Tianyou Mou, Siwen Wang, Andy Athawale, Luke E. K. Achenie, and Hongliang Xin. 2021. Interpretable machine learning of chemical bonding at solid surfaces. *J. Phys. Chem. Lett.* 12, 46 (2021), 11476–11487.

[27] Shingo Otsuka, Isao Kuwajima, Junko Hosoya, Yibin Xu, and Masayoshi Yamazaki. 2011. PoLyInfo: Polymer database for polymeric materials design. In *Proceedings of the International Conference on Emerging Intelligent Data and Web Technologies*. IEEE, 22–29.

[28] 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. *Adv. Neural Inf. Process. Syst.* 34 (2021).

[29] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2019. DropEdge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*.

[30] Elan Rosenfeld, Pradeep Kumar Ravikumar, and Andrej Risteski. 2021. The risks of invariant risk minimization. In *International Conference on Learning Representations*.

[31] 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.

[32] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. 2011. Weisfeiler-lehman graph kernels. *J. Mach. Learn. Res.* 12, 9 (2011).

[33] Qi Yuan, Mariagiulia Longo, Aaron W. Thornton, Neil B. McKeown, Bibiana Comesaña-Gándara, Johannes C. Jansen, Kim E. Jelfs. 2021. Imputation of missing gas permeability data for polymer membranes using machine learning. *Journal of Membrane Science* 627 (2021), 119207.

[34] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

[35] Daheng Wang, Tong Zhao, Nitesh V. Chawla, and Meng Jiang. 2021. Dynamic attributed graph prediction with conditional normalizing flows. In *Proceedings of the IEEE International Conference on Data Mining (ICDM'21)*. IEEE, 1385–1390.

[36] Yiwei Wang, Wei Wang, Yuxuan Liang, Yujun Cai, and Bryan Hooi. 2020. Graphcrop: Subgraph cropping for graph classification. arXiv:2009.10564. Retrieved from https://arxiv.org/abs/2009.10564

[37] Yiwei Wang, Wei Wang, Yuxuan Liang, Yujun Cai, and Bryan Hooi. 2021. Mixup for node and graph classification. In *Proceedings of the Web Conference*. 3663–3674.

[38] Yiwei Wang, Wei Wang, Yuxuan Liang, Yujun Cai, Juncheng Liu, and Bryan Hooi. 2020. Nodeaug: Semi-supervised node classification with data augmentation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 207–217.

[39] Qitian Wu, Hengrui Zhang, Junchi Yan, and David Wipf. 2022. Handling distribution shifts on graphs: An invariance perspective. In *International Conference on Learning Representations*. https://openreview.net/forum?id=FQOC5u-1egI

[40] Yingxin Wu, Xiang Wang, An Zhang, Xiangnan He, and Tat-Seng Chua. 2022. Discovering invariant rationales for graph neural networks. In *International Conference on Learning Representations*.

[41] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S. Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* 32, 1 (2020), 4–24.

[42] 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. *Chem. Sci.* 9, 2 (2018), 513–530.

[43] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks? In *International Conference on Learning Representations*.

[44] 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. ChemRxiv. (2021) DOI: 10.26434/chemrxiv-2021-p4g7z

[45] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *International Conference on Machine Learning*. PMLR, 40–48.

[46] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. Gnnexplainer: Generating explanations for graph neural networks. *Adv. Neural Inf. Process. Syst.* 32 (2019), 9240.

[47] Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. 2018. Hierarchical graph representation learning with differentiable pooling. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 4805–4815.

[48] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. *Adv. Neural Inf. Process. Syst.* 33 (2020), 5812–5823.

[49] Tong Zhao, Wei Jin, Yozen Liu, Yingheng Wang, Gang Liu, Stephan Günneman, Neil Shah, and Meng Jiang. 2023. Graph data augmentation for graph machine learning: A survey. *IEEE Data Engineering Bulletin* 47, 2 (2023), 140–168.

[50] Tong Zhao, Gang Liu, Daheng Wang, Wenhao Yu, and Meng Jiang. 2022. Learning from counterfactual links for link prediction. In *International Conference on Machine Learning*. PMLR, 26911–26926.

[51] Tong Zhao, Yozen Liu, Leonardo Neves, Oliver Woodford, Meng Jiang, and Neil Shah. 2021. Data augmentation for graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 11015–11023.

[52] Tong Zhao, Bo Ni, Wenhao Yu, Zhichun Guo, Neil Shah, and Meng Jiang. 2021. Action sequence augmentation for early graph-based anomaly detection. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2668–2678.

[53] Jiajun Zhou, Jie Shen, and Qi Xuan. 2020. Data augmentation for graph classification. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2341–2344.

[54] Qi Zhu, Natalia Ponomareva, Jiawei Han, and Bryan Perozzi. 2021. Shift-robust gnns: Overcoming the limitations of localized graph training data. *Adv. Neural Inf. Process. Syst.* 34 (2021), 27965–27977.

[55] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2021. Graph contrastive learning with adaptive augmentation. In *Proceedings of the Web Conference*. 2069–2080.

Received 20 June 2023; revised 6 October 2023; accepted 13 December 2023