Learning Invariant Graph Representations for Out-of-Distribution Generalization

Haoyang Li, Ziwei Zhang, Xin Wang, Wenwu Zhu
Tsinghua University
lihy18@mails.tsinghua.edu.cn, {zwzhang, xin_wang, wwzhu}@tsinghua.edu.cn

Abstract

Graph representation learning has shown effectiveness when testing and training graph data come from the same distribution, but most existing approaches fail to generalize under distribution shifts. Invariant learning, backed by the invariance principle from causality, can achieve guaranteed generalization under distribution shifts in theory and has shown great successes in practice. However, invariant learning for graphs under distribution shifts remains unexplored and challenging. To solve this problem, we propose Graph Invariant Learning (GIL) model capable of learning generalized graph representations under distribution shifts. Our proposed method can capture the invariant relationships between predictive graph structural information and labels in a mixture of latent environments through jointly optimizing three tailored modules. Specifically, we first design a GNN-based subgraph generator to identify invariant subgraphs. Then we use the variant subgraphs, i.e., complements of invariant subgraphs, to infer the latent environment labels. We further propose an invariant learning module to learn graph representations that can generalize to unknown test graphs. Theoretical justifications for our proposed method are also provided. Extensive experiments on both synthetic and real-world datasets demonstrate the superiority of our method against state-of-the-art baselines under distribution shifts for the graph classification task.

1 Introduction

Graph structured data is ubiquitous in the real world, e.g., social networks, biology networks, chemical molecules, etc. Graph representation learning, which encodes graphs into vectorized representations, has been the central topic in graph machine learning in the last decade. For example, graph neural networks (GNNs) design end-to-end learning schemes to extract useful graph information and are shown to be successful in a variety of applications.

Despite the enormous success, the existing approaches for learning graph representations heavily rely on the I.I.D. assumption, i.e., the testing and training graph data are independently drawn from an identical distribution. However, distribution shifts of graph data widely exist in real-world scenarios and are usually inevitable due to the uncontrollable underlying data generation mechanism. Most existing approaches fail to generalize to out-of-distribution (OOD) testing graph data. One critical bottleneck is that the existing methods ignore the invariant graph patterns and tend to rely on correlations that are variant for graphs from different environments. Therefore, it is of paramount significance to learn graph representations under distribution shifts and develop methods capable of out-of-distribution (OOD) generalization. Such studies are particularly critical for high-stake graph applications such as medical diagnosis, financial analysis, molecular prediction, etc.

In this work, we propose a brand new methodology to learn invariant graph representation under distribution shifts. Invariant learning, which aims to exploit the invariant relationships between
features and labels across different distributions while disregarding the variant spurious correlations, can provably achieve satisfactory OOD generalization under distribution shifts [8–10]. Though invariant learning has been studied for images and texts [8, 11], it remains largely unexplored in the literature of graph representation learning. However, invariant graph representation learning is non-trivial due to the following challenges. First, graph data usually comes from a mixture of latent environments without accurate environment labels, as shown in Figure 1. Since most invariant methods require multiple training environments with explicit environment labels, these existing methods cannot be directly applied to graphs. Second, the formation process of graphs is affected by the complex interaction of both invariant and variant patterns. How to identify the invariant patterns among latent environments is even more challenging. Last but not least, even after having obtained the environmental labels, how to design a theoretically grounded learning scheme to generate graph representations capable of OOD generalization remains largely unexplored. To tackle these challenges, in this paper, we propose Graph Invariant Learning method (GIL) which is able to capture invariant graph patterns in a mixture of latent environments and capable of OOD generalization under distribution shifts. As shown in Figure 2, our proposed method can capture the invariant relationships between predictive graph structural information and labels in a mixture of latent environments through jointly optimizing three mutually promoting modules, with each module tackling one aforementioned challenge. Specifically, in the invariant subgraph identification module, we design a GNN-based subgraph generator to identify potentially invariant subgraphs from the complex interaction between invariant and variant patterns. Then, we use the variant subgraphs, i.e., the complement of invariant subgraphs, to infer environment labels by clustering these environment-discriminative features. The variant subgraphs capture variant correlations under different distributions and therefore contain informative features to infer environment labels. Lastly, in the invariant learning module, we propose to optimize the maximal invariant subgraph generator criterion given the identified invariant subgraphs and inferred environments to generate graph representations capable of OOD generalization under distribution shifts. We theoretically show that the OOD generalization problem on graphs can be formulated as finding a maximal invariant subgraph generator of our GIL, and further prove that our GIL satisfies permutation invariance. We conduct extensive experiments on both synthetic graph datasets and real graph benchmarks for the graph classification task. The results show that the representations learned from GIL achieve substantial performance gains on the unseen OOD testing graphs compared with various state-of-the-art baselines.

Our contributions are summarized as follows.

- We propose a novel Graph Invariant Learning method (GIL) to learn invariant and OOD generalized graph representations under distribution shifts. To the best of our knowledge, we are the first to study invariant learning for graph representation learning under a mixture of latent environments.
- Our proposed method can automatically infer the environment label of graphs from a mixture of latent environments without supervision.
- We propose maximal invariant subgraph generator criterion to learn graph representations capable of OOD generalization under distribution shifts.
- We theoretically show that finding a maximal invariant subgraph generator of GIL can solve the OOD generalization problem. Extensive empirical results demonstrate the effectiveness of GIL on various synthetic and benchmark datasets under distribution shifts.

2 Notations and Problem Formulation

Notations. Let $G$ and $Y$ be the graph and label space. We consider a graph dataset $G = \{(G_i, Y_i)\}_{i=1}^{N}$ where $G_i \in G$ and $Y_i \in Y$. Following the OOD convention [11, 8, 9], we assume the dataset is collected from multiple training environments, i.e., $G = \{G_e\}_{e \in \text{supp}(\xi_e)}$, where $G^e = \{(G^e_i, Y^e_i)\}_{i=1}^{N}$ denotes the dataset from environment $e$, $\text{supp}(\xi_e)$ is the support of the environmental variable in the training data. We use $G$ and $Y$ to denote the random variables of graph and label, and $G^e$ and $Y^e$ to
Problem Formulation. We formulate the generalization under distribution shifts on graphs as:

Problem 1. Let \( E \) denote the random variable on indices of all possible environments. Our goal is to find an optimal predictor \( f^*(\cdot) : G \rightarrow \mathcal{Y} \) that performs well on all environments:

\[
\text{arg min}_{f} \sup_{e \in \text{supp}(E)} \mathcal{R}(f|e),
\]

where \( \mathcal{R}(f|e) = \mathbb{E}_{G,Y}[\ell(f(G), Y)] \) is the risk of the predictor \( f \) on the environment \( e \), and \( \ell(\cdot, \cdot) : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R} \) denotes a loss function. We further decompose \( f(\cdot) = w \circ h \), where \( h(\cdot) : G \rightarrow \mathbb{R}^d \) is the representation learning function, \( d \) is the dimensionality, and \( w(\cdot) : \mathbb{R}^d \rightarrow \mathcal{Y} \) is the classifier.

Note that \( \text{supp}(E_{tr}) \subset \text{supp}(E) \). Besides, distribution shifts indicate that \( P^e(G, Y) \neq P^{e'}(G, Y), e \in \text{supp}(E_{tr}), e' \in \text{supp}(E) \setminus \text{supp}(E_{tr}) \), i.e., the joint distribution of the graph and the corresponding label is different for training and testing graph data.

Problem 1 is difficult to be solved since \( \text{supp}(E) \) is unobserved or latent \([8, 9]\). In addition, for most graph datasets, we do not have access to accurate environment labels or environment partitions. Therefore, we focus on jointly inferring the environments of the graph dataset \( \mathcal{G} \) and achieving good OOD generalization performance under the inferred environments. The problem is formulated as:

Problem 2. Given a graph dataset \( \mathcal{G} \) collected from a mixture of latent environments but without environment labels, the task is to jointly infer graph environments \( E_{inf} \), i.e., \( \mathcal{G} = \{G^e\} \in \text{supp}(E_{inf}) \), and learn a graph predictor \( f^*(\cdot) \) in Problem 1 under the inferred environments \( E_{inf} \) to achieve good OOD generalization performance.

3 Method

In this section, we introduce our proposed method in detail, whose framework is shown in Figure 2. We first present the invariant subgraph identification module. Then, we infer environment labels by clustering the variant subgraphs. Next, we introduce the maximal invariant subgraph generator criterion to generate graph representations which can generalize to test graphs under distribution shifts. Lastly, we provide some discussions of our proposed method.
3.1 Invariant Subgraph Identification

We assume that each input graph $G \in \mathcal{G}$ has an invariant subgraph $G_I \subset G$ so that its relationship with the label is invariant across different environments. We refer to the rest of the graph, i.e., the complement of $G_I$, as the variant subgraph and denote it as $G_V$. $G_V$ represents the graph part whose relationship with the label is variant across different environments, e.g., spurious correlations. Therefore, the model will have a better OOD generalization ability if it can identify the invariant subgraph and only uses structural information from $G_I$.

We denote a generator to obtain the invariant subgraph as $G_I = \Phi(G)$. Following the invariant learning literature \cite{DBLP:conf/icml/SantoroPPGG18}, we make an assumption on $\Phi(G)$ as follows:

**Assumption 3.1.** Given $G$, there exists an optimal invariant subgraph generator $\Phi^*(G)$ satisfying:

a. Invariance property: $\forall e, e' \in \text{supp}(E), P^e(Y | \Phi^*(G)) = P^{e'}(Y | \Phi^*(G))$.

b. Sufficiency property: $Y = w^* (g^*(\Phi^*(G))) + \epsilon, \epsilon \perp G$, where $g^*(\cdot)$ denotes a representation learning function, $w^*$ is the classifier, $\perp$ indicates statistical independence, and $\epsilon$ is random noise.

The invariance assumption means that there exists a subgraph generator such that it can generate invariant subgraphs across different environments. The sufficiency assumption means that the generated invariant subgraphs should have sufficient predictive abilities in predicting the graph labels. Under this assumption, we instantiate $\Phi(\cdot)$ with learnable parameters. Consider an input graph instance $G$ with $n$ nodes. The corresponding adjacency matrix is denoted as $A = \{0, 1\}^{n \times n}$, where $A_{i,j} = 1$ represents that there is an edge between node $i$ and $j$, and $A_{i,j} = 0$ otherwise. To split the input graph $G$ into $G_I$ and $G_V$, a common strategy is to use a binary mask matrix $M = \{0, 1\}^{n \times n}$ on the adjacency matrix $A$. However, directly optimizing a discrete matrix $M$ is intractable as $G$ has exponentially many subgraph candidates \cite{DBLP:conf/iclr/ZitnikK18}. Besides, learning $M$ for each graph $G$ separately hinders the method from handling unseen test graphs \cite{DBLP:conf/iclr/ZitnikK18}. Therefore, we adopt a shared learnable GNN (denoted as $\text{GNN}^M$) to generate a soft mask matrix $M = \mathbb{R}^{n \times n}$ as follows:

$$M_{i,j} = Z_{i}^{(m)\top} \cdot Z_{j}^{(m)}, \quad Z^{(m)} = \text{GNN}^M(G),$$

where $Z^{(m)}$ is the node representation. Then, we obtain the invariant and variant subgraphs as:

$$A_I = \text{Top}_t(M \odot A), \quad A_V = A - A_I,$$

where $A_I$ and $A_V$ denotes the adjacency matrix of $G_I$ and $G_V$, respectively, $\odot$ means the element-wise matrix multiplication, and $\text{Top}_t(\cdot)$ selects the top $t$-percentage of elements with the largest values. The parameters of $\text{GNN}^M$ are trained on all available graphs to generate the corresponding $G_I$ and $G_V$. Using the inductive learning ability of GNNs, it can also be used to unseen test graphs, as opposed to directly optimizing mask matrices.

3.2 Environment Inference

After obtaining the invariant and variant subgraphs, we can infer the environment label $E_{\text{infer}}$. The intuition is that since the invariant subgraph captures invariant relationships between predictive graph structural information and labels, the variant subgraphs in turn capture variant correlations under different distributions, which are environment-discriminative features. Therefore, we can use the variant subgraphs to infer the latent environments. We adopt another GNN encoder, whose parameters are also shared among different graphs, to generate the representation of the variant subgraph $G_V$:

$$Z_V = \text{GNN}^V(G_V), \quad h_V = \text{READOUT}^V(Z_V),$$

where $\text{READOUT}$ is a permutation-invariant readout function that aggregates node-level representation $Z_V$ into graph-level representation $h_V$. The representation of all the variant subgraphs is denoted as $H = [h_V^1, ..., h_V^K]$. After obtaining $H$, we use an off-the-shelf clustering algorithm to infer the environment label $E_{\text{infer}}$. In this paper, we adopt k-means \cite{DBLP:journals/corr/FreyH07} as our clustering algorithm:

$$E_{\text{infer}} = \text{k-means}(H).$$

Using $E_{\text{infer}}$, we can partition the graph dataset into multiple training environments, i.e., $\mathcal{G}_{\text{train}} = \{\mathcal{G}^e\}_{e \in \text{supp}(E_{\text{infer}})}$. The environment inference module is purely unsupervised without needing any ground-truth environment labels.
3.3 Invariant Learning

After obtaining the inferred invariant subgraphs and environment labels, we propose the invariant learning module which can generate OOD generalized graph representations under distribution shifts.

Recall that both the invariant subgraph identification module and environment inference module heavily depend on the generator \( \Phi \). Therefore, we aim to learn the optimal generator \( \Phi^* \) in Assumption 3.1 by proposing and optimizing the maximal invariant subgraph generator criterion. First, following the invariant learning literature \cite{9}, we give the following definition.

**Definition 1.** The invariant subgraph generator set \( \mathcal{I} \) with respect to \( \mathcal{E} \) is defined as:

\[
\mathcal{I}_E = \{ \Phi(\cdot) : P^e(Y|\Phi(G)) = P^e(Y|\Phi(G)), e, e' \in \text{supp}(\mathcal{E}) \}.
\]  

(6)

Then, we show that the optimal generator \( \Phi^* \) satisfies the following theorem.

**Theorem 3.2.** A generator \( \Phi(G) \) is the optimal generator that satisfies Assumption 3.1 if and only if it is the maximal invariant subgraph generator, i.e.,

\[
\Phi^* = \arg \max_{\Phi \in \mathcal{I}_E} I(Y; \Phi(G)),
\]

(7)

where \( I(\cdot; \cdot) \) is the mutual information between the label and the generated subgraph.

The proof is provided in Appendix. Eq. (7) provides us an objective function to optimize the subgraph generator. However, directly solving Eq. (7) for a non-linear \( \Phi \) is difficult \cite{9}. Following the invariant learning literature \cite{9}, we transform Eq. (7) into an invariance regularizer:

\[
E_{e \in \text{supp}(E_{\text{in}, \text{f} e r})} R_e(f(G), Y; \theta) + \lambda \text{trace}(\text{Var} E_{\text{in}, \text{f} e r}(\nabla \theta R_e)),
\]

(8)

where \( f(\cdot) = w \circ g \circ \Phi, E_{\text{in}, \text{f} e r} \) is the inferred environment label, and \( \theta \) denotes all the learnable parameters. Recall that \( g(\cdot) \) is the representation learning function of the invariant subgraphs and \( w(\cdot) \) is the classifier. We instantiate \( g \) as another GNN as follows:

\[
Z_I = \text{GNN}^4(G_I), h_I = \text{READOUT}^4(Z_I).
\]

(9)

\( Z_I \) and \( h_I \) are the node-level and graph-level representations of invariant subgraph \( G_I \), respectively. \( w(\cdot) \) is instantiated as a multilayer perceptron followed by the softmax activation function. By optimizing Eq. (8), we can get our desired generator \( \Phi \) and the subgraph representation learning function \( g(\cdot) \), which collectively serve as our representation learning method \( h(\cdot) \), i.e., \( h = g \circ \Phi \).

3.4 Discussions

**Training Procedure.** We present the pseudocode of GIL in Appendix.

**Time Complexity.** The time complexity of our GIL is \( O(|E| d + |V| d^2) \), where \( |V| \) and \( |E| \) denotes the number of nodes and edges, respectively, and \( d \) is the dimensionality of the representations. Specifically, we adopt message-passing GNNs to instantiate our GNN components, which has a complexity of \( O(|E| d + |V| d^2) \). Since we only need to generate mask for the existing edges in graphs, the time complexity of generating invariant and variant subgraphs and further obtaining their representations is \( O(|E| d + |V| d^2) \). The time complexity of environment inference is \( O(|B| E_{\text{in}, \text{f} e r} |T^d|) \), where \( |B| \) is the batch size, \( T \) is the number of iterations for the k-means algorithm, and \( E_{\text{in}, \text{f} e r} \) denotes the number of inferred environments. The time complexity of the invariance regularizer is \( O(|E_{\text{in}, \text{f} e r}| d^2) \), as the number of parameters for most GNNs is \( O(d^2) \). Since \( |B|, |E_{\text{in}, \text{f} e r}|, \) and \( T \) are small constants, the overall time complexity of GIL is \( O(|E| d + |V| d^2) \). In comparison, the time complexity of other GNN-based graph representation methods is also \( O(|E| d + |V| d^2) \). Therefore, the time complexity of our proposed GIL is on par with the existing methods.

4 Theoretical Analysis

In this section, we theoretically analyze our GIL model by showing that the maximal invariant subgraph generator can achieve OOD optimal. The proofs are provided in Appendix.

**Theorem 4.1.** Let \( \Phi^* \) be the optimal invariant subgraph generator in Assumption 3.1 and denote the complement as \( G \setminus \Phi^*(G) \), i.e., the corresponding variant subgraph. Then, we can obtain the optimal predictor under distribution shifts, i.e., the solution to Problem 7 as follows:

\[
\arg \min_{w, g} w \circ g \circ \Phi^*(G) = \arg \min_f \sup_{e \in \text{supp}(\mathcal{E})} R(f|e),
\]

(10)
if the following conditions hold: (1) \( \Phi^*(G) \perp G \setminus \Phi^*(G) \); and (2) \( \forall \Phi \in \mathcal{I}_E, \exists e' \in \text{supp}(\mathcal{E}) \) such that \( P_e^r(G, Y) = P_e^r(\Phi(G), Y)P_e^r(G \setminus \Phi(G)) \) and \( P_e^r(\Phi(G)) = P_e^r(\Phi(G)) \).

The theorem shows that we can transform the OOD generalization problem into finding the optimal invariant subgraphs while maintaining the optimality.

We also prove that our GIL satisfies permutation invariance in Appendix.

## 5 Experiments

In this section, we evaluate the effectiveness of our GIL on both synthetic and real-world datasets.

### 5.1 Experimental Setup

#### Datasets.
We adopt one synthetic dataset with controllable ground-truth environments and four real-world benchmark datasets for the graph classification task.

- **SP-Motif**: Following [13, 16], we generate a synthetic dataset where each graph consists of one variant subgraph and one invariant subgraph, i.e., motif. The variant subgraph includes Tree, Ladder, and Wheel (denoted by \( V = 0, 1, 2 \), respectively) and the invariant subgraph includes Cycle, House, and Crane (denoted by \( I = 0, 1, 2 \)). The ground-truth label \( Y \) only depends on the invariant subgraph \( I \), which is sampled uniformly. The spurious correlation between \( V \) and \( Y \) is injected by controlling the variant subgraphs distribution as: \( P(V) = r \) if \( V = I \) and \( P(V) = (1 - r)/2 \) if \( V \neq I \). Intuitively, \( r \) controls the strength of the spurious correlation. We set \( r \) to different values in the testing and training set to simulate the distribution shifts.

- **MNIST-75sp** [17]: The task is to classify each graph that is converted from an image in MNIST [18] into the corresponding handwritten digit. Distribution shifts exist on node features by adding random noises in the testing set.

- **Graph-SST2** [19]: Each graph is converted from a text sequence. Graphs are split into different sets based on average node degrees to create distribution shifts.

- **Open Graph Benchmark (OGB)** [20]: We consider two datasets, MOLSIDER and MOLHIV. The default split separates structurally different molecules with different scaffolds into different subsets.

#### Baselines.
We compare our GIL with some representative state-of-the-art methods. The first group of these methods generates masks on graph structures to filter out important subgraphs using different GNNs, including Attention [2], Top-k Pool [21], SAGPool [22], and ASAP [23]. The second group is invariant learning methods, including standard ERM, GroupDRO [24], IRM [8], V-REx [25], GNNs, including Attention [2], Top-k Pool [21], SAGPool [22], and ASAP [23]. The second group of these methods generates masks on graph structures to filter out important subgraphs using different

#### Optimization and Hyper-parameters.
The adopted GNNs and READOUT functions including GNN\textsuperscript{M}, GNN\textsuperscript{V}, GNN\textsuperscript{I}, READOUT\textsuperscript{V}, and READOUT\textsuperscript{I} are listed in Appendix. The hyper-parameter \( \lambda \) in Eq. (5) is chosen from \( \{10^{-5}, 10^{-3}, 10^{-1}\} \). The number of clusters in Eq. (5) is chosen from \( \{2, 4\} \). They are tuned on the validation set. We report the mean results and standard deviations of five runs. More details on the datasets, baselines and implementations are in Appendix.

### 5.2 Experiments on SP-Motif

#### Settings.
To simulate different degrees of distribution shifts, we vary \( r \) in both the training and testing datasets. For the training set, we select \( r_{\text{train}} \) from \( \{1/3, 0.5, 0.6, 0.7, 0.8, 0.9\} \). A larger \( r_{\text{train}} \) indicates a higher spurious correlation between \( Y \) and \( G_V \) in the training set, while \( r_{\text{train}} = 1/3 \) implies that the training set is balanced without any spurious correlation. For the testing set, we consider two settings: (1) \( r_{\text{test}} = 1/3 \), which simulates that the invariant subgraphs and variant subgraphs are randomly attached without spurious correlations; (2) \( r_{\text{test}} = 0.2 \), which indicates that the testing set has reversed spurious correlations and thus is more challenging.

#### Quantitative Results.
The results are shown in Table 1. We have the following observations. Our proposed GIL model consistently and significantly outperforms the baselines and achieves the best performance on all settings. The results demonstrate that our proposed method can well handle graph distribution shifts and have a remarkable out-of-distribution generalization ability.
When the degree of distribution shift is relatively small, GNNs with different pooling methods to generate subgraphs generally report better results. On the other hand, when the degree of distribution shift is large, invariant baselines show more stable performance. Among them, DIR, which is a recently proposed invariant method specifically designed for graphs, is one competitive baseline. Nevertheless, our proposed method outperforms DIR by more than 3% in terms of the classification accuracy in most cases. GSAT achieves promising gains over the other baselines, but our GIL still performs better than GSAT. When $r_{test} = r_{train} = 1/3$, i.e., no distribution shifts, our proposed method also achieves the best results, indicating that learning invariant subgraphs is also beneficial.

**Analysis.** To analyze whether our proposed method can accurately capture the invariant subgraph, we compare GIL with baselines that also output subgraphs using the ground-truth invariant subgraphs. The evaluation metric is Precision@5. We report the results in Figure 5. The results show that GIL has a clear advantage in discovering invariant subgraphs under latent environments, while the other baselines cannot handle distribution shifts well.
Table 2: The graph classification results (%) on testing sets of the real-world datasets. We report the accuracy for MNIST-75sp and Graph-SST2, ROC-AUC for MOLSIDER and MOLHIV.

<table>
<thead>
<tr>
<th></th>
<th>MNIST-75sp</th>
<th>Graph-SST2</th>
<th>MOLSIDER</th>
<th>MOLHIV</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERM</td>
<td>73.26</td>
<td>57.37</td>
<td>62.04</td>
<td>14.37</td>
</tr>
<tr>
<td>Attention</td>
<td>64.44</td>
<td>81.44</td>
<td>60.53</td>
<td>15.37</td>
</tr>
<tr>
<td>Top-4 Pool</td>
<td>15.02</td>
<td>79.78</td>
<td>65.43</td>
<td>15.37</td>
</tr>
<tr>
<td>SAG Pool</td>
<td>15.34</td>
<td>80.24</td>
<td>65.32</td>
<td>15.37</td>
</tr>
<tr>
<td>ASAP</td>
<td>15.34</td>
<td>80.24</td>
<td>65.32</td>
<td>15.37</td>
</tr>
<tr>
<td>GroupPRO</td>
<td>15.34</td>
<td>80.24</td>
<td>65.32</td>
<td>15.37</td>
</tr>
<tr>
<td>BBM</td>
<td>16.75</td>
<td>57.10</td>
<td>74.67</td>
<td>15.37</td>
</tr>
<tr>
<td>V-REX</td>
<td>18.40</td>
<td>71.64</td>
<td>75.62</td>
<td>15.37</td>
</tr>
<tr>
<td>DBR</td>
<td>17.38</td>
<td>57.74</td>
<td>77.05</td>
<td>15.37</td>
</tr>
<tr>
<td>GSAT</td>
<td>20.24</td>
<td>60.82</td>
<td>76.47</td>
<td>15.37</td>
</tr>
<tr>
<td>GIL</td>
<td>21.94</td>
<td>83.44</td>
<td>79.08</td>
<td>15.37</td>
</tr>
</tbody>
</table>

Besides the quantitative evaluation, we plot a showcase from the testing set of SP-Motif ($r_{train} = 0.8$ and $r_{test} = 0.2$) in Figure 7. The figure shows that the learned invariant subgraph of our method is more accurate than baselines.

5.3 Experiments on Real-world Graphs

We further evaluate the effectiveness of our method on real-world graph datasets. The experimental results are presented in Table 2. Our GIL achieves the best performance on all four datasets, indicating that GIL can well handle distribution shifts on real-world graphs. For example, GIL increases the classification accuracy by 1.8% on MNIST-75sp and ROC-AUC by 2.0% on MOLHIV against the strongest baselines respectively. On MOLHIV, the results of most baselines are worse than ERM, indicating that they fail to achieve OOD generalization in this dataset. Besides, different datasets have different distribution shifts, e.g., Graph-SST2 has different node degrees, the distribution shift of MNIST-75sp is on node features, and OGB is split based on scaffold. Therefore, the results show that our proposed method can well handle diverse types of distribution shifts in real graph datasets.

For MOLHIV, besides adopting GIN as backbone (shown in Table 2), our method is also compatible with the other popular GNNs. We try using HIG and CIN as the backbone since these models are orthogonal to ours. Table 3 shows that our GIL can consistently improve these models.

In addition, we present some showcases of the learned invariant subgraph of the proposed GIL on both the train and test set of Graph-SST2. This dataset consists of sentences with positive/negative sentiments and is more understandable for humans. Figure 7 shows that our method can learn invariant subgraphs consistently focusing on the positive/negative words that are salient for sentiments and distinguishing invariant/variant parts under distribution shifts. The subgraph “the world’s best actors” identified by GIL has a predictive and invariant relationship with the positive sentiment label, while the subgraph “daniel auteuil” may reflect variant sentiments in different sentences. These results validate: (1) the invariant and variant subgraphs widely exist in real-world datasets, and (2) our GIL can well identify invariant subgraphs under distribution shifts and further make predictions with high accuracy based on the learned invariant subgraphs.

5.4 Analysis of Environment Inference

In our proposed model, all components are jointly optimized. To show that the environment inference module and invariant learning module can mutually enhance each other, we record the test accuracy and the Silhouette score, which is a commonly used evaluation metric for clustering, as the model is trained. The results on SP-Motif ($r_{train} = 0.8$, $r_{test} = 1/3$) are shown in Figure 4. We can observe that the test accuracy and the clustering performance improve synchronously over training. A plausible reason is that, as the training stage progresses, the invariant subgraph generator is optimized so that it can generate more informative invariant subgraphs and therefore improve the performance on the testing set. On the other hand, accurately discovering invariant subgraphs can also promote
identifying variant subgraphs, which capture the environment-discriminate features and better infer the latent environments. To verify that GIL can infer the environments accurately, we use t-SNE [29] to plot the discovered environments on a 2D-plane when the optimization is finished. Figure 5 shows that the variant subgraphs perfectly capture the environment-discriminate features. Notice that GIL achieves such results without needing any ground-truth environment label.

5.5 Hyper-parameter Sensitivity

We investigate the sensitivity of hyper-parameters of our method, including the number of environments $|E_{infer}|$, the invariance regularizer coefficient $\lambda$, and the size of the invariant subgraph mask $t$ in Eq. (3). For simplicity, we only report the results on SP-Motif ($r_{train} = 0.8$ and $r_{test} = 1/3$) and MNIST-75sp in Figure 8 while the results on other datasets show similar patterns.

First, the number of environments has a moderate impact on the model performance. For SP-Motif, the performance reaches a peak when $|E_{infer}| = 3$, showing that GIL achieves the best result when the number of environments matches the ground truth. For MNIST-75sp, the best number of environments is $|E_{infer}| = 2$. A plausible reason is that a large number of environments will bring difficulty in inferring the latent environment, leading to sub-optimal performance. Second, the coefficient $\lambda$ also has a slight influence on the performance, indicating that we need to properly balance the classification loss and the invariance regularizer term. Finally, a proper value of the mask size $t$ is important. A very large $t$ will result in too many edges in the invariant subgraph and bring in variant structures, while a small $t$ may let the invariant subgraph become too small to capture enough structural information. Although an appropriate choice of hyper-parameters can further improve the performance, our method is not very sensitive to hyper-parameters. Figure 8 shows that GIL can outperform the best baselines with a wide range of hyper-parameters choices.

6 Related Works

**Graph neural networks.** Recently, graph neural networks (GNNs) have shown enormous success in graph representation learning [1, 3], demonstrating their strength in various tasks [30, 36]. GNNs generally adopt a neighborhood aggregation (message passing) paradigm, i.e., the representations of nodes are iteratively updated by aggregating representations of their neighbors. The representation of the whole graph is summarized on node representations through the readout function (i.e., pooling) [5, 22]. However, most existing GNN models do not consider the out-of-distribution generalization ability [37] so that their performances can drop substantially on testing graphs with distribution shifts.

**Generalization of GNNs.** Early works [38–41] for analyzing the generalization ability of GNNs do not consider distribution shifts [37] [42, 43]. More recently, the generalization ability of GNNs under distribution shifts starts to receive research attention [44, 46] [45, 40]. [47] find that encoding task-specific non-linearities in the architecture or features can improve GNNs in extrapolating graph algorithmic tasks. [17] [48] [49] try to encourage GNNs to perform well on testing graphs with different sizes. Some works [50, 51] are proposed to deal with node-level tasks. EERM [52] studies the OOD generalization in node classification. However, little attention has been paid to learning graph-level representations under distribution shifts starts to receive research attention [44, 16, 45, 46]. [47] find that encoding task-specific non-linearities in the architecture or features can improve GNNs in extrapolating graph algorithmic tasks. [17] [48] [49] try to encourage GNNs to perform well on testing graphs with different sizes. Some works [50, 51] are proposed to deal with node-level tasks. EERM [52] studies the OOD generalization in node classification. However, little attention has been paid to learning graph-level representations under distribution shifts from the invariant learning perspective. One exception is the work DIR [16], which conducts interventions on graphs to create interventional distributions. However, performing causal intervention relies on strong assumptions [53] that could be violated and expensive to satisfy in practice [54]. GSAT [26] applies graph information bottleneck criteria for generalization, but its goal is mainly to build inherently interpretable GNNs.

**Invariant Learning.** Invariant learning aims to exploit the invariant relationships between features and labels across distribution shifts, while filtering out the variant spurious correlations. Backed by causal theory, invariant learning model can lead to OOD optimal models under some assumptions [11]
However, most existing methods heavily rely on multiple environments that have to be explicitly provided in the training dataset. Such annotation is not only prohibitively expensive for graphs, but also inherently problematic as the environment split could be inaccurate, rendering these invariant learning methods inapplicable. A few works study OOD generalization on latent environments in computer vision \cite{55,56} or raw feature data \cite{57}, which cannot be directly applied to graphs. In summary, how to learn invariant graph representations without explicit environment label under distribution shifts remains largely unexplored in the literature.

7 Conclusions

In this paper, we propose the graph invariant learning (GIL) model to tackle the problem of learning invariant graph representations under distribution shifts. Three tailored modules are jointly optimized to encourage the graph representations to capture the invariant relationships between predictive graph structural information and labels. Theoretical analysis and extensive experiments on both synthetic and real-world datasets demonstrate the superiority of GIL.

Acknowledgements

This work was supported in part by the National Key Research and Development Program of China No. 2020AAA0106300 and National Natural Science Foundation of China (No. 62250008, 62222209, 62102222, 62206149), China National Postdoctoral Program for Innovative Talents No. BX20220185, China Postdoctoral Science Foundation No. 2022M711813.

References


Qi Zhu, Natalia Ponomareva, Jiawei Han, and Bryan Perozzi. Shift-robust gnns: Overcoming the limitations of localized graph training data. Advances in Neural Information Processing Systems, 34, 2021.


Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes]

(b) Did you mention the license of the assets? [Yes]

(c) Did you include any new assets either in the supplemental material or as a URL? [N/A]

(d) Did you discuss whether and how consent was obtained from people whose data you’re using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]