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) [1–3] 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 [4]. 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 [5], financial analysis [6], molecular prediction [7], 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

[*]Corresponding authors

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.

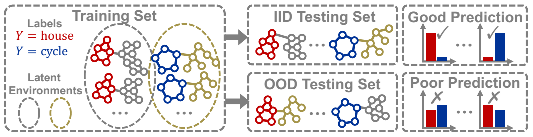


Figure 1: An example of distribution shifts under a mixture of latent environments, which leads to poor generalization.

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 $\mathbb{G}$ and $\mathbb{Y}$ be the graph and label space. We consider a graph dataset $\mathcal{G} = \{(G_i, Y_i)\}_{i=1}^{N}$ where $G_i \in \mathbb{G}$ and $Y_i \in \mathbb{Y}$. Following the OOD convention [11, 8, 9], we assume the dataset is collected from multiple training environments, i.e., $\mathcal{G} = \{\mathcal{G}^e\}_{e \in \mathrm{supp}(\mathcal{E}_{tr})}$, where $\mathcal{G}^e = \{(G_i^e, Y_i^e)\}_{i=1}^{N_e}$ denotes the dataset from environment e, $\mathrm{supp}(\mathcal{E}_{tr})$ 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

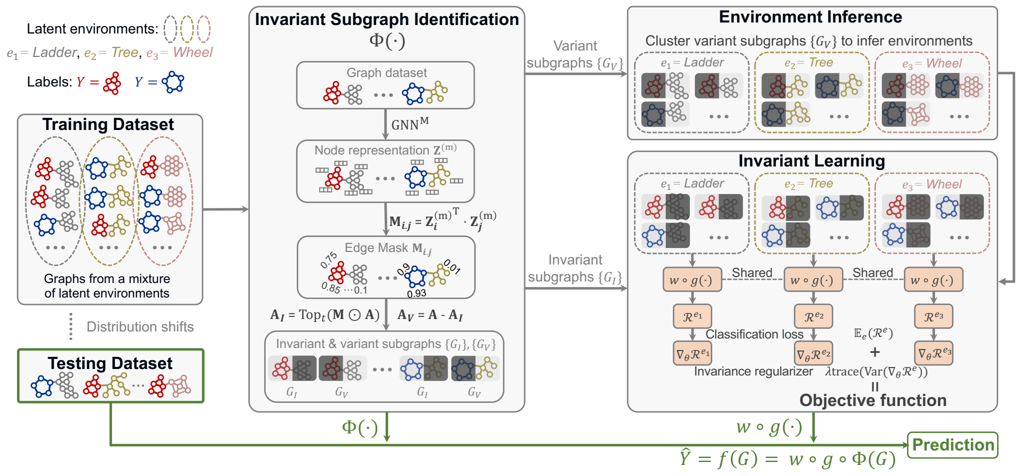

Latent environments:
e_1 = Ladder, e_2 = Tree, e_3 = Wheel
Labels: Y = Y =
Training Dataset
Graphs from a mixture of latent environments
Distribution shifts
Testing Dataset
Invariant Subgraph Identification $\Phi(\cdot)$
Graph dataset
GNN$^{\mathrm{M}}$
Node representation $\mathbf{Z}^{(\mathrm{m})}$
$\mathbf{M}_{ij} = \mathbf{Z}_i^{(\mathrm{m})^{\mathrm{T}}} \cdot \mathbf{Z}_j^{(\mathrm{m})}$
Edge Mask $\mathbf{M}_{ij}$
0.75
0.85
0.1
0.9
0.01
0.93
$\mathbf{A}_I = \mathrm{Top}_t(\mathbf{M} \odot \mathbf{A})$
$\mathbf{A}_V = \mathbf{A} - \mathbf{A}_I$
Invariant & variant subgraphs $\{G_I\}, \{G_V\}$
G_I G_V G_I G_V
Variant subgraphs $\{G_V\}$
Invariant subgraphs $\{G_I\}$
Environment Inference
Cluster variant subgraphs $\{G_V\}$ to infer environments
e_1 = Ladder
e_2 = Tree
e_3 = Wheel
Inferred environments $\mathcal{E}_{infer}$
Invariant Learning
$w \circ g(\cdot)$
Shared
$\mathcal{R}^{e_1}$ $\mathcal{R}^{e_2}$ $\mathcal{R}^{e_3}$
Classification loss
$\mathbb{E}_e(\mathcal{R}^e)$
$\nabla_\theta \mathcal{R}^{e_1}$ $\nabla_\theta \mathcal{R}^{e_2}$ $\nabla_\theta \mathcal{R}^{e_3}$
+
Invariance regularizer
$\lambda \mathrm{trace}(\mathrm{Var}(\nabla_\theta \mathcal{R}^e))$
Objective function
$\Phi(\cdot)$
$w \circ g(\cdot)$
Prediction
$\hat{Y} = f(G) = w \circ g \circ \Phi(G)$

Figure 2: The framework of **GIL** model. Our proposed method jointly optimizes three modules: (1) In the invariant subgraph identification module, a GNN-based subgraph generator $\Phi(\cdot)$ identifies the invariant subgraph G_I and the variant subgraph G_V. (2) The environment inference module uses the variant subgraphs $\{G_V\}$ to infer the latent environments by clustering the representations of $\{G_V\}$. (3) The invariant learning module jointly optimizes the invariant subgraph generator $\Phi(\cdot)$, the representation learning function $g(\cdot)$, and the classifier $w(\cdot)$. **Training stage** (shown by grey arrows): we back propagate with the objective function to update model parameters. **Testing stage** (shown by green arrows): we use the optimized model to make predictions.

specify random variables from environment e. The environment label for graphs is *unobserved* since it is prohibitively expensive to collect graph environment labels for most real scenarios.

Problem Formulation. We formulate the generalization under distribution shifts on graphs as:

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

$$f^*(\cdot) = \arg\min_f \sup_{e \in \mathrm{supp}(\mathcal{E})} \mathcal{R}(f|e), \tag{1}$$

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

Note that $\mathrm{supp}(\mathcal{E}_{tr}) \subset \mathrm{supp}(\mathcal{E})$. Besides, distribution shifts indicate that $P^e(\mathrm{G}, \mathrm{Y}) \neq P^{e'}(\mathrm{G}, \mathrm{Y}), e \in \mathrm{supp}(\mathcal{E}_{tr}), e' \in \mathrm{supp}(\mathcal{E}) \setminus \mathrm{supp}(\mathcal{E}_{tr})$, 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 $\mathrm{supp}(\mathcal{E})$ 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 $\mathcal{E}_{infer}$, i.e., $\mathcal{G} = \{\mathcal{G}^e\}_{e \in \mathrm{supp}(\mathcal{E}_{infer})}$, and learn a graph predictor $f^*(\cdot)$ in Problem 1 under the inferred environments $\mathcal{E}_{infer}$ 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 [12], we make an assumption on $\Phi(\mathrm{G})$ as follows:

Assumption 3.1. *Given* G*, there exists an optimal invariant subgraph generator* $\Phi^*(\mathrm{G})$ *satisfying:*
a. Invariance property*:* $\forall e, e' \in \mathrm{supp}(\mathcal{E}), P^e(\mathrm{Y}|\Phi^*(\mathrm{G})) = P^{e'}(\mathrm{Y}|\Phi^*(\mathrm{G}))$.
b. Sufficiency property*:* $\mathrm{Y} = w^*(g^*(\Phi^*(\mathrm{G}))) + \epsilon,\ \epsilon \perp \mathrm{G}$, *where* $g^*(\cdot)$ *denotes a representation learning function,* w^* *is the classifier,* $\perp$ *indicates statistical independence, and* ϵ *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 $\mathbf{A} = \{0,1\}^{n\times n}$, where $\mathbf{A}_{i,j} = 1$ represents that there exists an edge between node i and j, and $\mathbf{A}_{i,j} = 0$ otherwise. To split the input graph G into G_I and G_V, a common strategy is to use a binary mask matrix $\mathbf{M} = \{0,1\}^{n\times n}$ on the adjacency matrix $\mathbf{A}$. However, directly optimizing a discrete matrix $\mathbf{M}$ is intractable as G has exponentially many subgraph candidates [13]. Besides, learning $\mathbf{M}$ for each graph G separately hinders the method from handling unseen test graphs [14]. Therefore, we adopt a shared learnable GNN (denoted as $\mathrm{GNN}^{\mathbf{M}}$) to generate a soft mask matrix $\mathbf{M} = \mathbb{R}^{n\times n}$ as follows:

$$\mathbf{M}_{i,j} = \mathbf{Z}_i^{(m)\top} \cdot \mathbf{Z}_j^{(m)},\ \mathbf{Z}^{(m)} = \mathrm{GNN}^{\mathbf{M}}(G), \tag{2}$$

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

$$\mathbf{A}_I = \mathrm{Top}_t\left(\mathbf{M} \odot \mathbf{A}\right), \mathbf{A}_V = \mathbf{A} - \mathbf{A}_I, \tag{3}$$

where $\mathbf{A}_I$ and $\mathbf{A}_V$ denotes the adjacency matrix of G_I and G_V, respectively, $\odot$ means the element-wise matrix multiplication, and $\mathrm{Top}_t(\cdot)$ selects the top t-percentage of elements with the largest values. The parameters of $\mathrm{GNN}^{\mathbf{M}}$ 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 $\mathcal{E}_{infer}$. 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:

$$\mathbf{Z}_V = \mathrm{GNN}^{\mathbf{V}}(G_V),\ \mathbf{h}_V = \mathrm{READOUT}^{\mathbf{V}}(\mathbf{Z}_V), \tag{4}$$

where READOUT is a permutation-invariant readout function that aggregates node-level representation $\mathbf{Z}_V$ into graph-level representation $\mathbf{h}_V$. The representation of all the variant subgraphs is denoted as $\mathbf{H} = [\mathbf{h}_{V_1}, ..., \mathbf{h}_{V_N}]$. After obtaining $\mathbf{H}$, we use an off-the-shelf clustering algorithm to infer the environment label $\mathcal{E}_{infer}$. In this paper, we adopt k-means [15] as our clustering algorithm:

$$\mathcal{E}_{infer} = \text{k-means}(\mathbf{H}). \tag{5}$$

Using $\mathcal{E}_{infer}$, we can partition the graph dataset into multiple training environments, i.e., $\mathcal{G} = \{\mathcal{G}^e\}_{e\in \mathrm{supp}(\mathcal{E}_{infer})}$. 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 Φ. Therefore, we aim to learn the optimal generator Φ^* in Assumption 3.1 by proposing and optimizing the **maximal invariant subgraph generator** criterion. First, following the invariant learning literature [9], we give the following definition.

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

$$\mathcal{I}_{\mathcal{E}} = \{\Phi(\cdot) : P^e(\mathrm{Y}|\Phi(\mathrm{G})) = P^{e'}(\mathrm{Y}|\Phi(\mathrm{G})), e, e' \in \mathrm{supp}(\mathcal{E})\}. \tag{6}$$

Then, we show that the optimal generator Φ^* satisfies the following theorem.

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

$$\Phi^* = \arg\max_{\Phi \in \mathcal{I}_{\mathcal{E}}} I\left(\mathrm{Y}; \Phi(\mathrm{G})\right), \tag{7}$$

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 Φ is difficult [9]. Following the invariant learning literature [9], we transform Eq. (7) into an invariance regularizer:

$$\mathbb{E}_{e \in \mathrm{supp}(\mathcal{E}_{infer})} \mathcal{R}^e(f(\mathrm{G}), \mathrm{Y}; \theta) + \lambda \mathrm{trace}(\mathrm{Var}_{\mathcal{E}_{infer}}(\nabla_\theta \mathcal{R}^e)), \tag{8}$$

where $f(\cdot) = w \circ g \circ \Phi$, $\mathcal{E}_{infer}$ is the infered environment label, and θ 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:

$$\mathbf{Z}_I = \mathrm{GNN}^{\mathbf{I}}(G_I), \mathbf{h}_I = \mathrm{READOUT}^{\mathbf{I}}(\mathbf{Z}_I). \tag{9}$$

$\mathbf{Z}_I$ and $\mathbf{h}_I$ 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 Φ 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 $O(|\mathcal{B}||\mathcal{E}_{infer}|Td)$, where $|\mathcal{B}|$ is the batch size, T is the number of iterations for the k-means algorithm, and $|\mathcal{E}_{infer}|$ denotes the number of inferred environments. The time complexity of the invariance regularizer is $O(|\mathcal{E}_{infer}|d^2)$, as the number of parameters for most GNNs is $O(d^2)$. Since $|\mathcal{B}|$, $|\mathcal{E}_{infer}|$, 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 Φ^* be the optimal invariant subgraph generator in Assumption 3.1 and denote the complement as $\mathrm{G}\backslash\Phi^*(\mathrm{G})$, i.e., the corresponding variant subgraph. Then, we can obtain the optimal predictor under distribution shifts, i.e., the solution to Problem 1, as follows:*

$$\arg\min_{w,g} w \circ g \circ \Phi^*(\mathrm{G}) = \arg\min_{f} \sup_{e \in \mathrm{supp}(\mathcal{E})} \mathcal{R}(f|e), \tag{10}$$

if the following conditions hold: (1) $\Phi^*(\mathrm{G}) \perp \mathrm{G}\backslash\Phi^*(\mathrm{G})$*; and (2)* $\forall \Phi \in \mathcal{I}_{\mathcal{E}}$*,* $\exists\ e' \in \operatorname{supp}(\mathcal{E})$ *such that* $P^{e'}(\mathrm{G},\mathrm{Y}) = P^{e'}(\Phi(\mathrm{G}),\mathrm{Y})P^{e'}(\mathrm{G}\backslash\Phi(\mathrm{G}))$ *and* $P^{e'}(\Phi(\mathrm{G})) = P^{e}(\Phi(\mathrm{G}))$.

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], DIR [16]. We also consider a recent interpretable graph learning method GSAT [26]. For a fair comparison, we use the same GNN backbone as **GIL** for the baselines.

Optimization and Hyper-parameters. The adopted GNNs and READOUT functions including $\mathrm{GNN}^{\mathbf{M}}$, $\mathrm{GNN}^{\mathbf{V}}$, $\mathrm{GNN}^{\mathbf{I}}$, $\mathrm{READOUT}^{\mathbf{V}}$, and $\mathrm{READOUT}^{\mathbf{I}}$ are listed in Appendix. The hyper-parameter λ in Eq. (8) 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_{train} from $\{1/3, 0.5, 0.6, 0.7, 0.8, 0.9\}$. A larger r_{train} indicates a higher spurious correlation between Y and G_V in the training set, while $r_{train} = 1/3$ implies that the training set is balanced without any spurious correlation. For the testing set, we consider two settings: (1) $r_{test} = 1/3$, which simulates that the invariant subgraphs and variant subgraphs are randomly attached without spurious correlations; (2) $r_{test} = 0.2$, 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.

Table 1: The graph classification accuracy (%) on testing sets of the synthetic dataset SP-Motif. In each column, the boldfaced and the underlined score denotes the best and the second-best result, respectively. Numbers in the lower right corner denote standard deviations.

	Scenario 1: $r_{test} = 1/3$						**Scenario 2**: $r_{test} = 0.2$					
r_{train}	$r = 1/3$	$r = 0.5$	$r = 0.6$	$r = 0.7$	$r = 0.8$	$r = 0.9$	$r = 1/3$	$r = 0.5$	$r = 0.6$	$r = 0.7$	$r = 0.8$	$r = 0.9$
ERM	53.60$_{\pm 3.79}$	51.24$_{\pm 4.13}$	47.04$_{\pm 7.01}$	38.80$_{\pm 3.72}$	37.84$_{\pm 3.01}$	37.44$_{\pm 2.15}$	48.48$_{\pm 4.53}$	41.72$_{\pm 4.81}$	36.92$_{\pm 6.93}$	35.72$_{\pm 8,33}$	28.80$_{\pm 3.91}$	19.60$_{\pm 1.66}$
Attention	54.31$_{\pm 3.98}$	53.24$_{\pm 3.56}$	42.52$_{\pm 6.20}$	35.20$_{\pm 1.05}$	34.48$_{\pm 1.18}$	33.88$_{\pm 1.01}$	44.04$_{\pm 4.33}$	31.64$_{\pm 0.67}$	25.72$_{\pm 5.34}$	24.80$_{\pm 4.06}$	23.20$_{\pm 3.60}$	18.04$_{\pm 2.88}$
Top-k Pool	54.68$_{\pm 2.71}$	53.12$_{\pm 5.58}$	44.56$_{\pm 4.57}$	37.44$_{\pm 2.04}$	35.24$_{\pm 2.28}$	34.28$_{\pm 4.11}$	45.68$_{\pm 5.16}$	34.20$_{\pm 4.34}$	31.00$_{\pm 2.89}$	30.64$_{\pm 3.59}$	29.16$_{\pm 2.18}$	27.56$_{\pm 3.91}$
SAG Pool	54.08$_{\pm 3.66}$	52.60$_{\pm 3.52}$	44.68$_{\pm 5.25}$	37.68$_{\pm 4.03}$	34.28$_{\pm 1.82}$	32.72$_{\pm 1.83}$	44.36$_{\pm 6.09}$	38.64$_{\pm 3.02}$	31.36$_{\pm 4.40}$	32.84$_{\pm 1.86}$	28.72$_{\pm 3.11}$	26.60$_{\pm 5.37}$
ASAP	54.00$_{\pm 4.21}$	51.92$_{\pm 3.81}$	45.12$_{\pm 1.98}$	36.28$_{\pm 0.86}$	34.24$_{\pm 2.02}$	34.40$_{\pm 3.15}$	49.88$_{\pm 4.90}$	34.52$_{\pm 4.35}$	27.00$_{\pm 2.61}$	27.20$_{\pm 2.53}$	27.96$_{\pm 3.89}$	22.88$_{\pm 4.33}$
GroupDRO	53.20$_{\pm 4.91}$	51.40$_{\pm 4.35}$	48.32$_{\pm 5.35}$	39.12$_{\pm 4.27}$	38.40$_{\pm 2.76}$	37.64$_{\pm 1.69}$	52.68$_{\pm 4.04}$	43.68$_{\pm 4.05}$	31.92$_{\pm 6.84}$	34.36$_{\pm 8.41}$	28.88$_{\pm 5.14}$	20.32$_{\pm 1.64}$
IRM	52.00$_{\pm 2.34}$	50.60$_{\pm 3.54}$	47.84$_{\pm 6.95}$	38.80$_{\pm 3.72}$	39.84$_{\pm 3.21}$	39.00$_{\pm 3.98}$	50.24$_{\pm 6.73}$	41.60$_{\pm 4.75}$	35.24$_{\pm 5.35}$	34.92$_{\pm 8.03}$	29.44$_{\pm 5.47}$	21.84$_{\pm 3.57}$
V-REx	53.16$_{\pm 3.25}$	46.04$_{\pm 6.11}$	45.36$_{\pm 3.66}$	40.24$_{\pm 3.86}$	39.48$_{\pm 3.00}$	39.12$_{\pm 3.48}$	50.56$_{\pm 2.83}$	37.16$_{\pm 6.24}$	34.52$_{\pm 3.00}$	29.72$_{\pm 4.58}$	27.32$_{\pm 3.18}$	24.04$_{\pm 6.08}$
DIR	52.96$_{\pm 5.06}$	52.08$_{\pm 1.93}$	50.12$_{\pm 2.76}$	49.84$_{\pm 2.46}$	45.20$_{\pm 1.11}$	41.24$_{\pm 4.73}$	50.68$_{\pm 5.20}$	49.96$_{\pm 1.75}$	45.44$_{\pm 6.00}$	40.56$_{\pm 2.36}$	39.92$_{\pm 4.53}$	32.52$_{\pm 4.59}$
GSAT	53.67$_{\pm 3.65}$	53.34$_{\pm 4.08}$	51.54$_{\pm 3.78}$	50.12$_{\pm 3.29}$	45.83$_{\pm 4.01}$	44.22$_{\pm 5.57}$	51.36$_{\pm 4.21}$	50.48$_{\pm 3.98}$	46.93$_{\pm 5.03}$	43.55$_{\pm 3.67}$	40.35$_{\pm 4.21}$	33.87$_{\pm 5.19}$
GIL	**55.44**$_{\pm 3.11}$	**54.56**$_{\pm 3.02}$	**53.60**$_{\pm 4.82}$	**53.12**$_{\pm 2.18}$	**51.24**$_{\pm 3.88}$	**46.04**$_{\pm 3.51}$	**54.80**$_{\pm 3.93}$	**52.48**$_{\pm 4.41}$	**50.08**$_{\pm 5.47}$	**47.44**$_{\pm 2.87}$	**46.36**$_{\pm 3.80}$	**35.80**$_{\pm 5.03}$

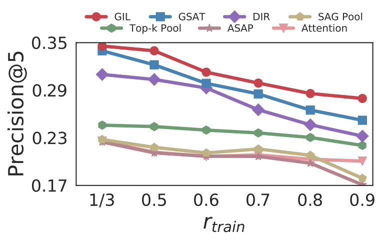


Figure 3: The results of discovering the ground-truth invariant subgraphs on SP-Motif ($r_{test} = 0.2$).

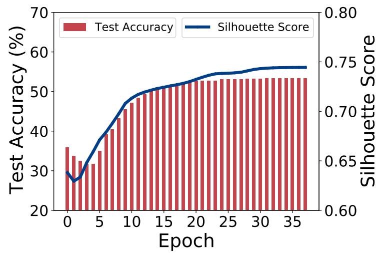


Figure 4: The test accuracy and the performance of environment inference over different training periods.

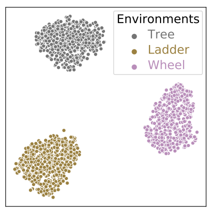
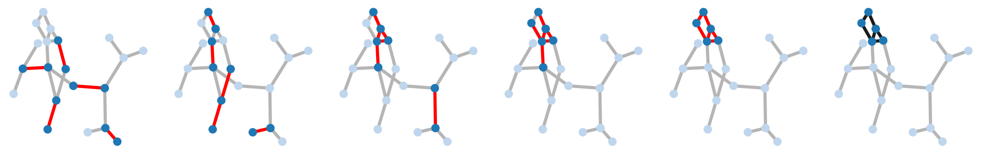


Figure 5: The environment inference results when training is finished.

As r_{train} grows larger, the performance of all the methods tends to decrease since there exists a larger degree of distribution shift. Nevertheless, our proposed method is able to maintain the most relatively stable performance. In fact, the performance gap between **GIL** and the baselines becomes more significant as the degree of distribution shift increases. For example, when $r_{test} = 1/3$, the accuracy of all baselines drops by more than 7% when r_{train} changes from 0.5 to 0.8, indicating their poor OOD generalization ability. In contrast, our method only has 3% performance drop.

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 3. 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.



(a) Top-k Pool (b) SAG Pool (c) DIR (d) GSAT (e) **GIL** (f) Ground Truth

Figure 6: Visualizations of the learned invariant subgraph for a showcase from the testing set of SP-Motif. In Figures (a)-(e), the red lines indicate the learned invariant subgraph, and the ground-truth is shown by the black lines in Figure (f).

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.

	MNIST-75sp	Graph-SST2	MOLSIDER	MOLHIV
ERM	14.94$\pm$3.27	81.44$\pm$0.59	57.57$\pm$1.56	76.20$\pm$1.14
Attention	16.44$\pm$3.78	81.57$\pm$0.71	56.99$\pm$0.54	75.84$\pm$1.33
Top-k Pool	15.02$\pm$3.08	79.78$\pm$1.35	60.63$\pm$1.52	73.01$\pm$1.65
SAG Pool	19.34$\pm$1.73	80.24$\pm$1.72	<u>61.29$\pm$1.31</u>	73.26$\pm$0.84
ASAP	15.14$\pm$3.58	81.57$\pm$0.84	55.77$\pm$1.34	73.81$\pm$1.17
GroupDRO	15.72$\pm$4.35	81.29$\pm$1.44	56.31$\pm$1.15	75.44$\pm$2.70
IRM	18.74$\pm$2.43	81.01$\pm$1.13	57.10$\pm$0.92	74.46$\pm$2.74
V-REx	18.40$\pm$1.12	81.76$\pm$0.08	57.76$\pm$0.78	75.62$\pm$0.79
DIR	17.38$\pm$3.52	<u>83.29$\pm$0.53</u>	57.74$\pm$1.63	<u>77.05$\pm$0.57</u>
GSAT	<u>20.12$\pm$1.35</u>	82.95$\pm$0.58	60.82$\pm$1.36	76.47$\pm$1.53
GIL	**21.94**$\pm$**0.38**	**83.44**$\pm$**0.37**	**63.50**$\pm$**0.57**	**79.08**$\pm$**0.54**

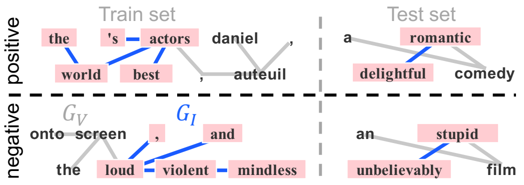


Figure 7: Four showcases of sentences with positive/negative sentiments of train/test sets on Graph-SST2 learned by our **GIL**. Blue edges indicate the learned invariant subgraphs, while the others are variant subgraphs.

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 6. 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.

Table 3: The test results with different backbones.

CIN (Rank #8)	**GIL (CIN Backbone)**	HIG (Rank #2)	PAS+FPs (Rank #1)	**GIL (HIG Backbone)**
80.94$\pm$0.57	**81.15**$\pm$**0.46**	84.03$\pm$0.21	84.20$\pm$0.15	**84.23**$\pm$**0.25**

For MOLHIV, besides adopting GIN [3] as backbone (shown in Table 2), our method is also compatible with the other popular GNNs. We try using HIG and CIN [27] (Rank #2 and #8 on the MOLHIV leaderboard[2]) 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 by consistently focusing on the positive/negative words that are salient for sentiments and distinguishing invariant/variant parts under distribution shifts. For example, 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 [28], 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

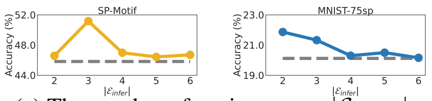
[2]`https://ogb.stanford.edu/docs/leader_graphprop/#ogbg-molhiv`

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.

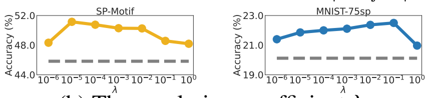
5.5 Hyper-parameter Sensitivity

We investigate the sensitivity of hyper-parameters of our method, including the number of environments $|\mathcal{E}_{infer}|$, the invariance regularizer coefficient λ, 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.

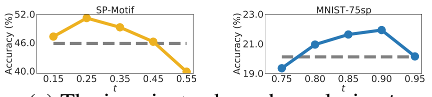
First, the number of environments has a moderate impact on the model performance. For SP-Motif, the performance reaches a peak when $|\mathcal{E}_{infer}| = 3$, showing that **GIL** achieves the best result when the number of environments matches the ground truth. For MNIST-75sp, the best number of environments is $|\mathcal{E}_{infer}| = 2$. 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 λ 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.



(a) The number of environments $|\mathcal{E}_{infer}|$.



(b) The regularizer coefficient λ.



(c) The invariant subgraph mask size t.

Figure 8: The impact of different hyper-parameters. Yellow and blue lines denote the results of **GIL** and grey dashed lines are the best results of all baselines.

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) [3, 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, 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, 8, 9]. 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 [55, 56] or raw feature data [57], which cannot be directly applied to graphs. In summary, how to learn invariant graph representations without explicit environment label under distribution shits 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

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

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

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

[4] Yoshua Bengio, Tristan Deleu, Nasim Rahaman, Rosemary Ke, Sébastien Lachapelle, Olexa Bilaniuk, Anirudh Goyal, and Christopher Pal. A meta-transfer objective for learning to disentangle causal mechanisms. *International Conference on Learning Representations*, 2019.

[5] Yang Li, Buyue Qian, Xianli Zhang, and Hui Liu. Graph neural network-based diagnosis prediction. *Big Data*, 8(5):379–390, 2020.

[6] Yiying Yang, Zhongyu Wei, Qin Chen, and Libo Wu. Using external knowledge for financial event prediction based on graph neural networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2161–2164, 2019.

[7] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

[8] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

[9] Masanori Koyama and Shoichiro Yamaguchi. Out-of-distribution generalization with maximal invariant predictor. *arXiv preprint arXiv:2008.01883*, 2020.

[10] Kartik Ahuja, Ethan Caballero, Dinghuai Zhang, Yoshua Bengio, Ioannis Mitliagkas, and Irina Rish. Invariance principle meets information bottleneck for out-of-distribution generalization. *Neural Information Processing Systems (NeurIPS)*, 2021.

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

[12] Mateo Rojas-Carulla, Bernhard Schölkopf, Richard Turner, and Jonas Peters. Invariant models for causal transfer learning. *The Journal of Machine Learning Research*, 19(1):1309–1342, 2018.

[13] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32:9240, 2019.

[14] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network. *Advances in Neural Information Processing Systems*, 33, 2020.

[15] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1):100–108, 1979.

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

[17] Boris Knyazev, Graham W Taylor, and Mohamed Amer. Understanding attention and generalization in graph neural networks. *Advances in Neural Information Processing Systems*, 32: 4202–4212, 2019.

[18] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[19] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. Explainability in graph neural networks: A taxonomic survey. *arXiv preprint arXiv:2012.15445*, 2020.

[20] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Neural Information Processing Systems (NeurIPS)*, 2020.

[21] Hongyang Gao and Shuiwang Ji. Graph u-nets. In *international conference on machine learning*, pages 2083–2092. PMLR, 2019.

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

[23] Ekagra Ranjan, Soumya Sanyal, and Partha Talukdar. Asap: Adaptive structure aware pooling for learning hierarchical graph representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5470–5477, 2020.

[24] Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *arXiv preprint arXiv:1911.08731*, 2019.

[25] David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Dinghuai Zhang, Remi Le Priol, and Aaron Courville. Out-of-distribution generalization via risk extrapolation (rex). In *International Conference on Machine Learning*, pages 5815–5826. PMLR, 2021.

[26] Siqi Miao, Mia Liu, and Pan Li. Interpretable and generalizable graph learning via stochastic attention mechanism. In *ICML*, 2022.

[27] Cristian Bodnar, Fabrizio Frasca, Nina Otter, Yuguang Wang, Pietro Lio, Guido F Montufar, and Michael Bronstein. Weisfeiler and lehman go cellular: Cw networks. *Neural Information Processing Systems*, 2021.

[28] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[29] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[30] Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu. Disentangled graph convolutional networks. In *International conference on machine learning*, pages 4212–4221. PMLR, 2019.

[31] Haoyang Li, Xin Wang, Ziwei Zhang, Jianxin Ma, Peng Cui, and Wenwu Zhu. Intention-aware sequential recommendation with structured intent transition. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[32] Haoyang Li, Xin Wang, Ziwei Zhang, Zehuan Yuan, Hang Li, and Wenwu Zhu. Disentangled contrastive learning on graphs. *Advances in Neural Information Processing Systems*, 34: 21872–21884, 2021.

[33] Xin Wang, Hong Chen, Yuwei Zhou, Jianxin Ma, and Wenwu Zhu. Disentangled representation learning for recommendation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[34] Haoyang Li, Peng Cui, Chengxi Zang, Tianyang Zhang, Wenwu Zhu, and Yishi Lin. Fates of microscopic social ecosystems: Keep alive or dead? In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 668–676, 2019.

[35] Haoyang Li, Ziwei Zhang, Xin Wang, and Wenwu Zhu. Disentangled graph contrastive learning with independence promotion. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[36] Yijian Qin, Xin Wang, Peng Cui, and Wenwu Zhu. Gqnas: Graph q network for neural architecture search. In *2021 IEEE International Conference on Data Mining (ICDM)*, pages 1288–1293. IEEE, 2021.

[37] Haoyang Li, Xin Wang, Ziwei Zhang, and Wenwu Zhu. Out-of-distribution generalization on graphs: A survey. *arXiv preprint arXiv:2202.07987*, 2022.

[38] Renjie Liao, Raquel Urtasun, and Richard Zemel. A pac-bayesian approach to generalization bounds for graph neural networks. In *International Conference on Learning Representations*, 2020.

[39] Vikas Garg, Stefanie Jegelka, and Tommi Jaakkola. Generalization and representational limits of graph neural networks. In *International Conference on Machine Learning*, pages 3419–3430. PMLR, 2020.

[40] Saurabh Verma and Zhi-Li Zhang. Stability and generalization of graph convolutional neural networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1539–1548, 2019.

[41] Franco Scarselli, Ah Chung Tsoi, and Markus Hagenbuchner. The vapnik–chervonenkis dimension of graph and recursive neural networks. *Neural Networks*, 108:248–259, 2018.

[42] Huaxiu Yao, Caroline Choi, Yoonho Lee, Pang Wei Koh, and Chelsea Finn. Wild-time: A benchmark of in-the-wild distribution shift over time. In *Proceedings of the Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[43] Zeyang Zhang, Ziwei Zhang, Xin Wang, and Wenwu Zhu. Learning to solve travelling salesman problem with hardness-adaptive curriculum. In *36th AAAI Conference on Artificial Intelligence (AAAI)*, 2022.

[44] Haoyang Li, Xin Wang, Ziwei Zhang, and Wenwu Zhu. Ood-gnn: Out-of-distribution generalized graph neural network. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[45] Yijian Qin, Xin Wang, Ziwei Zhang, Pengtao Xie, and Wenwu Zhu. Graph neural architecture search under distribution shifts. In *International Conference on Machine Learning*, pages 18083–18095. PMLR, 2022.

[46] Zeyang Zhang, Xin Wang, Ziwei Zhang, Haoyang Li, Zhou Qin, and Wenwu Zhu. Dynamic graph neural networks under spatio-temporal distribution shift. In *Thirty-Sixth Conference on Neural Information Processing Systems*, 2022.

[47] Keyulu Xu, Mozhi Zhang, Jingling Li, Simon S Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. How neural networks extrapolate: From feedforward to graph neural networks. In *International Conference on Learning Representations*, 2021.

[48] Gilad Yehudai, Ethan Fetaya, Eli Meirom, Gal Chechik, and Haggai Maron. From local structures to size generalization in graph neural networks. In *International Conference on Machine Learning*, pages 11975–11986. PMLR, 2021.

[49] Beatrice Bevilacqua, Yangze Zhou, and Bruno Ribeiro. Size-invariant graph representations for graph classification extrapolations. In *Proceedings of the 38th International Conference on Machine Learning*, pages 837–851, 2021.

[50] 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.

[51] Shaohua Fan, Xiao Wang, Chuan Shi, Kun Kuang, Nian Liu, and Bai Wang. Debiased graph neural networks with agnostic label selection bias. *IEEE transactions on neural networks and learning systems*, 2022.

[52] Qitian Wu, Hengrui Zhang, Junchi Yan, and David Wipf. Handling distribution shifts on graphs: An invariance perspective. In *International Conference on Learning Representations (ICLR)*, 2022.

[53] Miguel A Hernán and James M Robins. Causal inference, 2010.

[54] Tan Wang, Chang Zhou, Qianru Sun, and Hanwang Zhang. Causal attention for unbiased visual recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3091–3100, 2021.

[55] Toshihiko Matsuura and Tatsuya Harada. Domain generalization using a mixture of multiple latent domains. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 11749–11756, 2020.

[56] Elliot Creager, Jörn-Henrik Jacobsen, and Richard Zemel. Environment inference for invariant learning. In *International Conference on Machine Learning*, pages 2189–2200. PMLR, 2021.

[57] Jiashuo Liu, Zheyuan Hu, Peng Cui, Bo Li, and Zheyan Shen. Heterogeneous risk minimization. In *International Conference on Machine Learning*. PMLR, 2021.

Learning Invariant Graph Representations for Out-of-Distribution Generalization
(Appendix)

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

A Notations

We summarize the key notations and the corresponding descriptions in Table 1.

Table 1: Notations.

Notation	Description
N	The number of graphs in the dataset
$\mathbb{G}, \mathbb{Y}$	The graph space and the label space
G, Y	A random variable of graph and label
G, Y	An instance of graph and label
$G_I = \Phi(G)$	An instance of the invariant subgraph and the invariant subgraph generator
Φ^*	The optimal invariant subgraph generator
$G_V = G \backslash G_I$	An instance of the variant subgraph
$\mathbf{A}_I / \mathbf{A}_V$	The adjacency matrix of the invariant/variant subgraph
$\mathbf{Z}_I / \mathbf{Z}_V$	The node-level representations of the invariant/variant subgraph
$\mathbf{h}_I / \mathbf{h}_V$	The graph-level representations of the invariant/variant subgraph
$\mathcal{E} / \mathcal{E}_{tr}$	A random variable on indices of all/training environments
$\mathcal{E}_{infer}$	A random variable on indices of the inferred environments
e	An instance of environment
f	The predictor from $\mathbb{G}$ to $\mathbb{Y}$
w	The classifier from $\mathbb{R}^d$ to $\mathbb{Y}$
h	The representation learning function from $\mathbb{G}$ to $\mathbb{R}^d$
g	The representation learning function for invariant subgraphs
$\mathcal{I}_{\mathcal{E}}$	The invariant subgraph generator set with respect to $\mathcal{E}$
ℓ	The loss function

B Training Procedure

To show the training procedure, we present the pseudocode of **GIL** in Algorithm 1.

C Explanations of Assumption

In the main paper, we denote the invariant subgraph generator as $\Phi(\cdot)$ and make the following assumption on $\Phi(\mathrm{G})$:

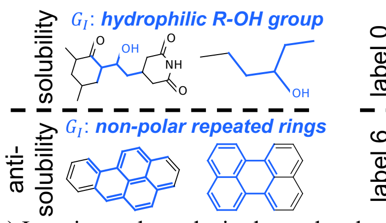
Assumption 3.1. *Given* G*, there exists an optimal invariant subgraph generator* $\Phi^*(\mathrm{G})$ *satisfying:*
a. Invariance property*:* $\forall e, e' \in \mathrm{supp}(\mathcal{E}), P^e(\mathrm{Y}|\Phi^*(\mathrm{G})) = P^{e'}(\mathrm{Y}|\Phi^*(\mathrm{G}))$.
b. Sufficiency property*:* $\mathrm{Y} = w^*(g^*(\Phi^*(\mathrm{G}))) + \epsilon,\ \epsilon \perp \mathrm{G}$, *where* $g^*(\cdot)$ *denotes a representation learning function,* w^* *is the classifier,* $\perp$ *indicates statistical independence, and* ϵ *is random noise.*

Algorithm 1 The training procedure of our **GIL**.

```
Input: Graph dataset $\mathcal{G} = \{(G_i, Y_i)\}_{i=1}^{N}$
Output: An optimized predictor $f(\cdot) : \mathbb{G} \to \mathbb{Y}$
 1: for sampled minibatch $\mathcal{B}$ of graph dataset $\mathcal{G}$ do
 2:     for each graph instance $(G, Y) \in \mathcal{B}$ do
 3:         Generate the mask matrix $\mathbf{M}$ with the shared learnable $\mathrm{GNN}^{\mathbf{M}}$ by Eq. (2).
 4:         Obtain the invariant subgraph $G_I$ and the variant subgraph $G_V$ by Eq. (3).
 5:         Generate representations $\mathbf{h}_V$ of variant subgraph $G_V$ by Eq. (4).
 6:         Generate representations $\mathbf{h}_I$ of invariant subgraph $G_I$ by Eq. (9).
 7:     end for
 8:     Infer environments $\mathcal{E}_{infer}$ with clustering representations of variant subgraphs $\mathbf{H}$ by Eq. (5).
 9:     Calculate the objective function by Eq. (8).
10:     Update model parameters using back propagation.
11: end for
```

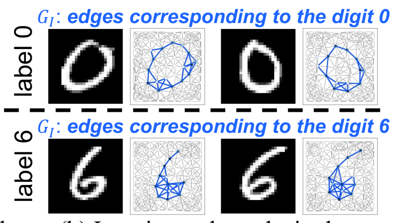
The invariance assumption means that there exists a subgraph generator such that it can generate invariant subgraphs across different environments. The sufficiency assumption means that the generated invariant subgraphs should have sufficient predictive abilities in predicting the graph labels. We make this assumption following the literature, e.g, [1–4].

To better illustrate this assumption is commonly satisfied, we provide real-world showcases in Figure 1. For molecule graphs from [3], invariant subgraph G_I (indicated by blue edges) represents "hydrophilic R-OH group"/"non-polar repeated ring structures", whose relationship with the label solubility/anti-solubility is truly predictive and invariant across different environments. And variant subgraph G_V denotes carbon structure or scaffold [4]. For superpixel graphs from [2], G_I and G_V represent the edges corresponding to the digit itself and other edges from the background, respectively.



G_I: hydrophilic R-OH group
solubility
O
OH
O
NH
O
OH
G_I: non-polar repeated rings
anti-solubility

(a) Invariant subgraphs in the molecules with solubility and anti-solubility.



G_I: edges corresponding to the digit 0
label 0
G_I: edges corresponding to the digit 6
label 6

(b) Invariant subgraphs in the superpixel graphs of digits 0 and 6.

Figure 1: Graph examples in real-world scenarios that include invariant G_I (blue edges) and variant G_V (gray edges) subgraphs.

D Proofs

In this section, we provide the proofs of Theorem 3.2 and 4.1. We also theoretically analyze that our **GIL** satisfies permutation invariance in Section D.3.

D.1 Proof of Theorem 3.2

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

$$\Phi^* = \arg \max_{\Phi \in \mathcal{I}_\mathcal{E}} I\left(\mathrm{Y}; \Phi(\mathrm{G})\right), \tag{1}$$

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

Proof. Denote $\hat{\Phi} = \arg\max_{\Phi \in \mathcal{I}_{\mathcal{E}}} I\left(\mathrm{Y}; \Phi(\mathrm{G})\right)$. From the invariance property of Assumption 3.1, $\Phi^* \in \mathcal{I}_{\mathcal{E}}$. Therefore, we prove the theorem by showing that $I(\mathrm{Y}; \hat{\Phi}(\mathrm{G})) \leq I(\mathrm{Y}; \Phi^*(\mathrm{G}))$ and consequently, $\hat{\Phi} = \Phi^*$.

To show the inequality, we use the functional representation lemma [5], which states that for any random variables X_1 and X_2, there exists a random variable X_3 independent of X_1 such that X_2 can be represented as a function of X_1 and X_3. So for $\Phi^*(\mathrm{G})$ and $\hat{\Phi}(\mathrm{G})$, there exists $\Phi'(\mathrm{G})$ satisfying that $\Phi'(\mathrm{G}) \perp \Phi^*(\mathrm{G})$ and $\hat{\Phi}(\mathrm{G}) = \gamma\left(\Phi^*(\mathrm{G}), \Phi'(\mathrm{G})\right)$, where $\gamma(\cdot)$ is a function. Then, we can derive that:

$$\begin{aligned} I(\mathrm{Y}; \hat{\Phi}(\mathrm{G})) &= I\left(\mathrm{Y}; \gamma\left(\Phi^*(\mathrm{G}), \Phi'(\mathrm{G})\right)\right) \\ &\leq I\left(\mathrm{Y}; \Phi^*(\mathrm{G}), \Phi'(\mathrm{G})\right) \\ &= I\left(w^*(g^*(\Phi^*(\mathrm{G}))); \Phi^*(\mathrm{G}), \Phi'(\mathrm{G})\right) \\ &= I\left(w^*(g^*(\Phi^*(\mathrm{G}))); \Phi^*(\mathrm{G})\right) = I\left(\mathrm{Y}; \Phi^*(\mathrm{G})\right), \end{aligned} \tag{2}$$

which finishes the proof. □

D.2 Proof of Theorem 4.1

Theorem 4.1. *Let Φ^* be the optimal invariant subgraph generator in Assumption 3.1 and denote the complement as $\mathrm{G}\backslash\Phi^*(\mathrm{G})$, i.e., the corresponding variant subgraph. Then, we can obtain the optimal predictor under distribution shifts, i.e., the solution to Problem 1, as follows:*

$$\arg\min_{w,g} w \circ g \circ \Phi^*(\mathrm{G}) = \arg\min_{f} \sup_{e \in \mathrm{supp}(\mathcal{E})} \mathcal{R}(f|e), \tag{3}$$

if the following conditions hold: (1) $\Phi^*(\mathrm{G}) \perp \mathrm{G}\backslash\Phi^*(\mathrm{G})$; and (2) $\forall \Phi \in \mathcal{I}_{\mathcal{E}}$, $\exists\ e' \in \mathrm{supp}(\mathcal{E})$ such that $P^{e'}(\mathrm{G},\mathrm{Y}) = P^{e'}(\Phi(\mathrm{G}),\mathrm{Y})P^{e'}(\mathrm{G}\backslash\Phi(\mathrm{G}))$ and $P^{e'}(\Phi(\mathrm{G})) = P^{e}(\Phi(\mathrm{G}))$.

Proof. Denote the function to obtain the complement of invariant subgraph as $\Psi(\mathrm{G}) = \mathrm{G}\backslash\Phi(\mathrm{G})$ and $\Psi^*(\mathrm{G}) = \mathrm{G}\backslash\Phi^*(\mathrm{G})$. By assumption, $\Phi^*(\mathrm{G}) \perp \Psi^*(\mathrm{G})$. Further denote $\hat{f} = \arg\min_{w,g} w \circ g \circ \Phi^*(\mathrm{G})$. By Assumption 3.1, we have

$$\hat{f}(\mathrm{G}) = w^* \circ g^* \circ \Phi^*(\mathrm{G}). \tag{4}$$

To show that $\hat{f}$ is f^*, our proof strategy is to show that $\forall e \in \mathrm{supp}(\mathcal{E})$, for any possible f, $\mathcal{R}(\hat{f}|e) \leq \mathcal{R}(f|e')$ and therefore $\sup_{e\in\mathrm{supp}(\mathcal{E})} \mathcal{R}(\hat{f}|e) \leq \sup_{e\in\mathrm{supp}(\mathcal{E})} \mathcal{R}(f|e)$.

To show the inequality, we have:

$$\mathcal{R}(\hat{f}|e) = \mathbb{E}^{e}_{\mathrm{G},\mathrm{Y}}[\ell(\hat{f}(\mathrm{G}),\mathrm{Y})] = \sum_{\mathrm{G},\mathrm{Y}} P^{e}(\mathrm{G},\mathrm{Y})\ell(\hat{f}(\mathrm{G}),\mathrm{Y}) \tag{5}$$

$$= \sum_{\Psi^*(\mathrm{G})} P^{e}(\Psi^*(\mathrm{G})) \sum_{\Phi^*(\mathrm{G}),\mathrm{Y}} P^{e}(\Phi^*(\mathrm{G}),\mathrm{Y})\ell\left(w^*(g^*(\Phi^*(\mathrm{G}))),\mathrm{Y}\right) \tag{6}$$

$$= \sum_{\Phi^*(\mathrm{G}),\mathrm{Y}} P^{e}(\Phi^*(\mathrm{G}),\mathrm{Y})\ell(w^*(g^*(\Phi^*(\mathrm{G}))),\mathrm{Y}) \tag{7}$$

$$\leq \sum_{\Phi(\mathrm{G}),\mathrm{Y}} P^{e}(\Phi(\mathrm{G}),\mathrm{Y})\ell(w(g(\Phi(\mathrm{G}))),\mathrm{Y}) \tag{8}$$

$$= \sum_{\Psi(\mathrm{G})} P^{e'}(\Psi(\mathrm{G})) \sum_{\Phi(\mathrm{G}),\mathrm{Y}} P^{e}(\Phi(\mathrm{G}),\mathrm{Y})\ell(w(g(\Phi(\mathrm{G}))),\mathrm{Y}) \tag{9}$$

$$= \sum_{\Psi(\mathrm{G})} \sum_{\Phi(\mathrm{G}),\mathrm{Y}} P^{e'}(\Phi(\mathrm{G}),\mathrm{Y})P^{e'}(\Psi(\mathrm{G}))\ell(w(g(\Phi(\mathrm{G}))),\mathrm{Y}) \tag{10}$$

$$= \sum_{\mathrm{G},\mathrm{Y}} P^{e'}(\mathrm{G},\mathrm{Y})\ell(f(\mathrm{G}),\mathrm{Y}) = \mathbb{E}^{e'}_{\mathrm{G},\mathrm{Y}}[\ell(f(\mathrm{G}),\mathrm{Y})] = \mathcal{R}(f|e'). \tag{11}$$

□

D.3 Proof of Permutation-invariance of GIL

Theorem D.3. *Our proposed **GIL** model is permutation-invariant if* $\mathrm{GNN}^{\mathbf{M}}$ *and* $\mathrm{GNN}^{\mathbf{I}}$ *are permutation-equivariant and* $\mathrm{READOUT}^{\mathbf{I}}$ *is permutation-invariant.*

Proof. The theorem is straight-forward from the compositionality of permutation-equivariant and invariant functions. Concretely, recall our learned model $f = w \circ g \circ \Phi$ is formulated as:

$$\begin{aligned} \Phi &: \mathbf{A}_I = \mathrm{Top}_t\left(\mathbf{M} \odot \mathbf{A}\right), \mathbf{M}_{i,j} = \mathbf{Z}_i^{(m)^\top} \cdot \mathbf{Z}_j^{(m)}, \mathbf{Z}^{(m)} = \mathrm{GNN}^{\mathbf{M}}(G). \\ g &: \mathbf{h}_I = \mathrm{READOUT}^{\mathbf{I}}(\mathbf{Z}_I), \mathbf{Z}_I = \mathrm{GNN}^{\mathbf{I}}(G_I) \\ w &: \hat{Y} = \mathrm{MLP}(\mathbf{h}_I). \end{aligned} \tag{12}$$

Consider $G' = \pi(G)$, where π is a permutation of nodes. We denote all variables for G' with a prime symbol in the top right corner. Since $\mathrm{GNN}^{\mathbf{M}}$ is permutation-equivariant, we have:

$$\mathbf{Z}^{(m)\prime} = \mathrm{GNN}^{\mathbf{M}}(G') = \mathrm{GNN}^{\mathbf{M}}(\pi(G)) = \pi(\mathrm{GNN}^{\mathbf{M}}(G)) = \pi(\mathbf{Z}^{(m)}). \tag{13}$$

Since the inner product matrix and $\mathrm{Top}_t(\cdot)$ are also equivariant with respect to permutations, we easily have:

$$\begin{gathered} \mathbf{M}'_{i,j} = \mathbf{Z}_i^{(m)\prime^\top} \cdot \mathbf{Z}_j^{(m)\prime} = \pi(\mathbf{Z}_i^{(m)})^\top \cdot \pi(\mathbf{Z}_j^{(m)}) = \pi(\mathbf{Z}_i^{(m)^\top} \cdot \mathbf{Z}_j^{(m)}) = \pi(\mathbf{M}_{i,j}) \\ \mathbf{A}'_I = \mathrm{Top}_t(\mathbf{M}' \odot \mathbf{A}') = \mathrm{Top}_t(\pi(\mathbf{M}) \odot \pi(\mathbf{A})) = \pi(\mathrm{Top}_t(\mathbf{M} \odot \mathbf{A})) = \pi(\mathbf{A}_I) \end{gathered} \tag{14}$$

Since $\mathrm{GNN}^{\mathbf{I}}$ are permutation-equivariant and $\mathrm{READOUT}^{\mathbf{I}}$ is permutation-invariant, we have:

$$\begin{gathered} \mathbf{Z}'_I = \mathrm{GNN}^{\mathbf{I}}(G'_I) = \mathrm{GNN}^{\mathbf{I}}(\pi(G_I)) = \pi(\mathrm{GNN}^{\mathbf{I}}(G_I)) = \pi(\mathbf{Z}_I) \\ \mathbf{h}'_I = \mathrm{READOUT}^{\mathbf{I}}(\mathbf{Z}'_I) = \mathrm{READOUT}^{\mathbf{I}}(\pi(\mathbf{Z}_I)) = \mathbf{h}_I. \end{gathered} \tag{15}$$

Therefore, we have $\hat{Y}' = \mathrm{MLP}(\mathbf{h}'_I) = \mathrm{MLP}(\mathbf{h}_I) = \hat{Y}$, i.e., f is permutation-invariant. □

The theorem shows that while learning invariant graph representations under distribution shifts, our proposed method naturally holds permutation-invariance as other GNNs.

E Experimental Details

E.1 Datasets

Table 2: The statistics of the datasets. #Graphs(Train/Val/Test) is the number of graphs in the training/validation/testing set of the dataset. #Classes is the number of classes. Average #Nodes/#Edges are the average number of nodes and edges in the graph of the dataset, respectively.

	SP-Motif	**MNIST-75sp**	**Graph-SST2**	**OGBG-MOLSIDER**	**OGBG-MOLHIV**
#Graphs(Train/Val/Test)	1,500/500/500	5,000/1,000/1,000	28,327/3,147/12,305	1,141/143/143	32,901/4,113/4,113
#Classes	3	10	2	2	2
Avg #nodes	26.7	66.8	13.7	33.6	25.5
Avg #edges	36.7	600.2	25.3	35.4	27.5

We adopt one synthetic dataset with controllable ground-truth environments and four real-world benchmark datasets for the graph classification task. The statistics of these datasets are provided in Table 2.

- **SP-Motif**: Following [6, 2], 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. Besides, we inject a spurious correlation between V and Y by controlling the variant subgraphs distribution as:

$$P(V) = \left\{ \begin{array}{cl} r, & \text{if } V = I \\ (1-r)/2, & \text{if } V \neq I \end{array} \right. . \tag{16}$$

Intuitively, r controls the strength of the spurious correlation between V and Y. We set r to different values in the testing and training set to simulate the distribution shifts, i.e., $r_{train} \in \{1/3, 0.5, 0.6, 0.7, 0.8, 0.9\}$ and $r_{test} \in \{1/3, 0.2\}$. For this dataset, we adopt random node features and constant edge weights.

- **MNIST-75sp** [7]: Each graph is converted from an image in MNIST [8] using superpixels [9]. We sample 7,000 images to build our dataset. The nodes are superpixels, and the edges are calculated by the spatial distance between nodes. The node features are set as the super-pixel coordinates and intensity. The task is to classify each graph into the corresponding handwritten digit labeled from 0 to 9. To simulate distribution shifts with respect to graph features, we follow [7] and generate testing graphs by colorizing images, i.e., adding two more channels and adding independent Gaussian noise, drawn from $\mathcal{N}(0, 0.6)$, to each channel.
- **Graph-SST2** [10]: Each graph is converted from a text sequence, where nodes represent words, edges indicate relations between words, and label is the sentence sentiment. Graphs are split into different sets according to their average node degree to create distribution shifts. We use constant edge weights and filter out the graphs with edges less than three. The node features are initialized by the pre-trained BERT word embedding [11].
- **Open Graph Benchmark (OGB)** [4]: We consider two graph property prediction datasets with distribution shifts, i.e., OGBG-MOLSIDER and OGBG-MOLHIV. The task is to predict the target molecular properties. We adopt the default scaffold splitting procedure, i.e., splitting the graphs based on their two-dimensional structural frameworks. Note that this scaffold splitting strategy aims to separate structurally different molecules into different subsets, which provides a more realistic and challenging scenario for testing graph out-of-distribution generalization.

The real-world datasets are publicly available as follows:

- **MNIST-75sp**: `http://yann.lecun.com/exdb/mnist/` with license unspecified
- **Graph-SST2**: `https://github.com/divelab/DIG/tree/main/dig/xgraph/datasets` with GPL-3.0 License
- **Open Graph Benchmark (OGB)**: `https://ogb.stanford.edu/docs/graphprop/` with MIT License

E.2 GNN Configurations

We summarize the backbone GNN for $\mathrm{GNN}^{\mathbf{M}}$, $\mathrm{GNN}^{\mathbf{V}}$, $\mathrm{GNN}^{\mathbf{I}}$ and readout function $\mathrm{READOUT}^{\mathbf{V}}$, $\mathrm{READOUT}^{\mathbf{I}}$ in Table 3. These settings are set the same as [2] for a fair comparison. For $\mathrm{GNN}^{\mathbf{M}}$, the number of layers is 2. $\mathrm{GNN}^{\mathbf{V}}$ and $\mathrm{GNN}^{\mathbf{I}}$ adopt shared parameters, and the number of layers is 4. The dimensionality of the graph-level and node-level representations d is 300 for OGB, 128 for Graph-SST2, and 32 for other datasets.

Table 3: The backbone GNNs and global pooling function of each dataset.

	SP-Motif	MNIST-75sp	Graph-SST2	OGBG-MOLSIDER	OGBG-MOLHIV
$\mathrm{GNN}^{\mathbf{M}}$/$\mathrm{GNN}^{\mathbf{V}}$/$\mathrm{GNN}^{\mathbf{I}}$ Backbone	Local Extremum GNN [12]	k-GNNs [13]	ARMA [14]	GIN + Virtual nodes [15, 4]	GIN + Virtual nodes [15, 4]
$\mathrm{READOUT}^{\mathbf{I}}$/$\mathrm{READOUT}^{\mathbf{V}}$	Mean Pooling	Max Pooling	Mean Pooling	Add Pooling	Add Pooling

E.3 Baselines

We provide detailed descriptions and links to code repository of baselines in our experiments as follows:

- **ERM**: We use ERM to denote the backbone GNN models listed in Table 3, which are trained with the standard empirical risk minimizing.

- **Attention**[1] [16]: For this baseline, we replace the default layers of $\text{GNN}^{\mathbf{M}}$ into graph attention layers. $\text{GNN}^{\mathbf{I}}$ and $\text{GNN}^{\mathbf{V}}$ are kept the same as ERM, which adopt the default layers shown in Table 3.
- **Top-k Pool**[2] [17]: This method implements the regular global top-k pooling operation on graph data. It selects a subset of important nodes to enable high-level feature encoding and receptive field enlargement. We add this pooling layer after the last layer of $\text{GNN}^{\mathbf{M}}$.
- **SAG Pool**[3] [18]: It exploits the self-attention mechanism to distinguish between the nodes that should be neglected and the nodes that should be chosen genearting the subgraph. Thanks to the self-attention mechanism which uses graph convolutions to calculate attention scores, node features and graph topology are jointly considered. We add this pooling layer after the last layer of $\text{GNN}^{\mathbf{M}}$.
- **ASAP**[4] [12]: It adopts self-attention with a modified GNN formulation to identify the importance of nodes in the graph and learns a sparse soft cluster assignment for nodes at each layer to effectively pool the subgraphs.
- **GroupDRO**[5] [19]: It seeks to optimize the worst-performance over a distribution set to achieve OOD generalization performance.
- **IRM**[6] [20]: It is a representative invariant learning method, seeking to find data representations or features for which the optimal predictor is invariant across all environments. We conduct random environment partitions on the input graph datasets for training.
- **V-REx**[7] [21]: This method is proven to be able to recover the causal mechanisms of the targets and is robust to distribution shifts. Since this method relies on the explicit environment labels that are unavailable for the graph datasets in a mixture of latent environments, we conduct random environment partitions on the input graph datasets during the training stage.
- **DIR**[8] [2]: It conducts interventions on graphs to create interventional distributions and improve generalization.
- **GSAT**[9] [22]: It aims to build inherently interpretable GNNs and expects GNNs to be more generalizable by penalizing the amount of information from the input data.

E.4 Additional Details of Optimization and Hyper-parameters

The number of epochs for optimizing our proposed method and baselines is set to 100. We adopt Stochastic Gradient Descent (SGD) for the optimization on Graph-SST2 and OGB datasets (the batch size is 32), and Gradient Descent (GD) for SP-Motif and MNIST-75sp, following the setting in [2] for a fair comparison. Each model is evaluated on the provided validation set for OGB or a holdout in-distribution validation set for the other datasets for each epoch. We adopt an early stopping strategy, i.e., stop training if the performance on the validation set does not improve for 5 epochs. Since we focus on graph classification tasks, we use the cross-entropy loss as the loss function ℓ. The activation function is ReLU [23]. The evaluation metric is ROC-AUC for OGB datasets and accuracy for the others. In the invariant subgraph indentification module, the invariant subgraph generator selects $t \times |E|$ edges for each graph, i.e., Top_t in Eq. (3), to generate the invariant subgraph. For a fair comparison, we uniformally set the hyper-parameter t for our method and baselines as 0.25, 0.9, 0.6, and 0.8 on SP-Motif, MNIST-75sp, Graph-SST2, and two OGB datasets, respectively. The hyper-parameter λ is chosen from $\{10^{-5}, 10^{-3}, 10^{-1}\}$ and the number of environments $|\mathcal{E}_{infer}|$ is chosen from $[2, 4]$ based on the results of the validation set. The selected λ and $|\mathcal{E}_{infer}|$ are reported in Table 4.

[1] `https://github.com/PetarV-/GAT` with MIT License

[2] `https://github.com/HongyangGao/Graph-U-Nets` with GPL-3.0 License

[3] `https://github.com/inyeoplee77/SAGPool` with license unspecified

[4] `https://github.com/malllabiisc/ASAP` with Apache-2.0 License

[5] `https://github.com/kohpangwei/group_DRO` with license unspecified

[6] `https://github.com/facebookresearch/InvariantRiskMinimization` with Attribution-NonCommercial 4.0 International License

[7] `https://github.com/capybaralet/REx_code_release` with license unspecified

[8] `https://github.com/wuyxin/dir-gnn` with MIT License

[9] `https://github.com/Graph-COM/GSAT` with MIT License

Table 4: The chosen hyper-parameters of λ and $|\mathcal{E}_{infer}|$ on each dataset.

	SP-Motif	MNIST-75sp	Graph-SST2	OGBG-MOLSIDER	OGBG-MOLHIV
λ	10^{-5}	10^{-5}	10^{-1}	10^{-3}	10^{-3}
$\|\mathcal{E}_{infer}\|$	3	2	2	2	2

E.5 Additional Details on Silhouette Score

Silhouette score [24], a commonly used evaluation metric for clustering, is defined as the mean Silhouette coefficient over all samples. The Silhouette coefficient is calculated using the mean intra-cluster distance (denoted as d_i) and the mean nearest-cluster distance (denoted as d_n) for each sample. The Silhouette coefficient for a sample is $(d_n - d_i)/max(d_i, d_n)$. Therefore, Silhouette score falls within the range $[-1, 1]$. A silhouette score close to 1 means that the clusters become dense and nicely separated. The score close to 0 means that clusters are overlapping. And the score of smaller than 0 means that data belonging to clusters may be wrong/incorrect.

E.6 Hardware and Software Configurations

We conduct the experiments with:

- Operating System: Ubuntu 18.04.1 LTS
- CPU: Intel(R) Xeon(R) CPU E5-2699 v4@2.20GHz
- GPU: NVIDIA GeForce GTX TITAN X with 12GB of Memory
- Software: Python 3.6.5; NumPy 1.19.2; PyTorch 1.10.1; PyTorch Geometric 2.0.3 [25].

E.7 Additional Experiments on Environment Inference

E.7.1 Experiments with Ground-truth Environments

We further conduct empirical analyses to investigate the performances of our model with the ground-truth environments. We compare our original model (**GIL**) with the model directly using the ground-truth environments (termed as **GIL** w. GT Env.) on the synthetic SP-Motif dataset ($r_{test} = 1/3$). The results in Table 5 show that the performance of using the inferred environments by our model and the ground-truth environments are comparable, even under different strengths of distribution shifts. The results are also expected since our inferred latent environments are largely aligned with the ground-truth labels, as shown in Figure 5 of the main paper. We think it would be interesting to conduct more explorations for real-world graphs when their environment labels are available.

Table 5: The accuracy (%) on SP-Motif ($r_{test} = 1/3$) when directly adopting the ground-truth environments (**GIL** w. GT Env.) compared with the original model (**GIL**).

r_{train}	$r = 1/3$	$r = 0.5$	$r = 0.6$	$r = 0.7$	$r = 0.8$	$r = 0.9$
GIL	55.44±3.11	54.56±3.02	53.60±4.82	53.12±2.18	51.24±3.88	46.04±3.51
GIL w. GT Env.	55.42±2.98	54.63±3.10	53.58±4.67	53.18±3.02	51.01±4.02	46.23±3.16

E.7.2 Experiments with Different Clustering Algorithms

As discussed in the main paper, we adopt the k-means clustering algorithm [26] to infer the environment labels. In addition, we explore another popular clustering algorithm [27] (termed as convex clustering) to infer the environment labels. The results on the synthetic SP-Motif dataset ($r_{test} = 1/3$) are shown in Table 6. These results indicate that the clustering algorithm could have a slight influence on the model performance and overall our model is not sensitive to the choice for clustering algorithm. Our proposed model does not rely on specific clustering algorithm to infer the environment labels and can also be compatible with other clustering algorithms.

Table 6: Environment inference with different clustering algorithms on the SP-Motif ($r_{test} = 1/3$).

r_{train}	$r = 1/3$	$r = 0.5$	$r = 0.6$	$r = 0.7$	$r = 0.8$	$r = 0.9$
GIL (k-means)	55.44±3.11	54.56±3.02	53.60±4.82	53.12±2.18	51.24±3.88	46.04±3.51
GIL (convex clustering)	55.21±2.45	53.60±4.74	54.01±5.13	53.43±1.94	50.12±4.15	47.01±2.54

E.8 Sensitivity of READOUT Functions and GNN Architectures

We conduct the sensitivity analysis on the READOUT functions and GNN architectures in Table 7. The results show that the choices of READOUT functions and GNN architectures have a slight influence on the performances. Overall, our model is not very sensitive to their choices and can be compatible with most common READOUT functions and GNN backbones.

Table 7: The performance (ROC-AUC, %) with different READOUT functions and GNN architectures.

	MOLSIDER	MOLHIV
GIN + add pooling	63.50±0.57	79.08±0.54
GIN + max pooling	63.37±0.72	**79.10±0.42**
GIN + mean pooling	61.91±0.75	78.16±0.47
GCN + add pooling	62.31±1.12	77.23±0.61
GCN + max pooling	**63.68±0.91**	77.61±0.59
GCN + mean pooling	61.33±0.45	76.98±0.36

References

[1] Mateo Rojas-Carulla, Bernhard Schölkopf, Richard Turner, and Jonas Peters. Invariant models for causal transfer learning. *The Journal of Machine Learning Research*, 19(1):1309–1342, 2018.

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

[3] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28, 2015.

[4] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Neural Information Processing Systems (NeurIPS)*, 2020.

[5] Abbas El Gamal and Young-Han Kim. *Network information theory*. Cambridge university press, 2011.

[6] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32:9240, 2019.

[7] Boris Knyazev, Graham W Taylor, and Mohamed Amer. Understanding attention and generalization in graph neural networks. *Advances in Neural Information Processing Systems*, 32: 4202–4212, 2019.

[8] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[9] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012.

[10] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. Explainability in graph neural networks: A taxonomic survey. *arXiv preprint arXiv:2012.15445*, 2020.

[11] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019.

[12] Ekagra Ranjan, Soumya Sanyal, and Partha Talukdar. Asap: Adaptive structure aware pooling for learning hierarchical graph representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5470–5477, 2020.

[13] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4602–4609, 2019.

[14] Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. Graph neural networks with convolutional arma filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

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

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

[17] Hongyang Gao and Shuiwang Ji. Graph u-nets. In *international conference on machine learning*, pages 2083–2092. PMLR, 2019.

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

[19] Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *arXiv preprint arXiv:1911.08731*, 2019.

[20] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

[21] David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Dinghuai Zhang, Remi Le Priol, and Aaron Courville. Out-of-distribution generalization via risk extrapolation (rex). In *International Conference on Machine Learning*, pages 5815–5826. PMLR, 2021.

[22] Siqi Miao, Mia Liu, and Pan Li. Interpretable and generalizable graph learning via stochastic attention mechanism. In *ICML*, 2022.

[23] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[24] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[25] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[26] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1):100–108, 1979.

[27] Danial Lashkari and Polina Golland. Convex clustering with exemplar-based models. *Advances in neural information processing systems*, 20, 2007.