Generalizing Graph Neural Networks on Out-Of-Distribution Graphs

Shaohua Fan, Xiao Wang, *Member, IEEE,* Chuan Shi†, *Senior Member, IEEE*, Peng Cui, *Senior Member, IEEE,* and Bai Wang

Abstract—Graph Neural Networks (GNNs) are proposed without considering the agnostic distribution shifts between training graphs and testing graphs, inducing the degeneration of the generalization ability of GNNs in Out-Of-Distribution (OOD) settings. The fundamental reason for such degeneration is that most GNNs are developed based on the I.I.D hypothesis. In such a setting, GNNs tend to exploit subtle statistical correlations existing in the training set for predictions, even though it is a spurious correlation. This learning mechanism inherits from the common characteristics of machine learning approaches. However, such spurious correlations may change in the wild testing environments, leading to the failure of GNNs. Therefore, eliminating the impact of spurious correlations is crucial for stable GNN models. To this end, in this paper, we argue that the spurious correlation exists among subgraph-level units and analyze the degeneration of GNN in causal view. Based on the causal view analysis, we propose a general causal representation framework for stable GNN, called StableGNN. The main idea of this framework is to extract high-level representations from raw graph data first and resort to the distinguishing ability of causal inference to help the model get rid of spurious correlations. Particularly, to extract meaningful high-level representations, we exploit a differentiable graph pooling layer to extract subgraph-based representations by an end-to-end manner. Furthermore, inspired by the confounder balancing techniques from causal inference, based on the learned high-level representations, we propose a causal variable distinguishing regularizer to correct the biased training distribution by learning a set of sample weights. Hence, GNNs would concentrate more on the true connection between discriminative substructures and labels. Extensive experiments are conducted on both synthetic datasets with various distribution shift degrees and eight real-world OOD graph datasets. The results well verify that the proposed model StableGNN not only outperforms the state-of-the-arts but also provides a flexible framework to enhance existing GNNs. In addition, the interpretability experiments validate that StableGNN could leverage causal structures for predictions. The source code is available at https://github.com/googlebaba/StableGNN.

Index Terms—Graph Neural Networks, Out-Of-Distribution Generalization, Causal Representation Learning, Stable Learning.

1 Introduction

Graph Neural Networks (GNNs) are powerful deep learning algorithms on graphs with various applications [2], [3], [4], [5]. One major category of applications is the predictive task over entire graphs, i.e., graph-level tasks, such as molecular graph property prediction [6], [7], [8], scene graph classification [9], and social network category classification [8], [10], etc. The success could be attributed to the non-linear modeling capability of GNNs, which extracts useful information from raw graph data and encodes them into the graph representation in a data-driven fashion.

The basic learning diagram of existing GNNs is to learn the parameters of GNNs from the training graphs, and then make predictions on unseen testing graphs. The most fundamental assumption to guarantee the success of such a learning diagram is the I.I.D. hypothesis, i.e., training and testing graphs are independently sampled from the identical distribution [11]. However, in reality, such a hypothesis is hardly satisfied due to the uncontrollable generation mechanism of real data, such as data selection biases, confounder factors or other peculiarities [12], [13], [14], [15]. The testing distribution may incur uncontrolled and unknown shifts from the training distribution, called Out-Of-Distribution (OOD) shifts [16], [17], which makes most GNN models fail to make stable predictions. As reported by OGB benchmark [6], GNN methods will occur a degeneration of 5.66% to 20% points when splitting the datasets according to the OOD settings, i.e., splitting structurally different graphs into training and testing sets.

Essentially, for general machine learning models, when there incurs a distribution shift, the main reason for the degeneration of accuracy is the spurious correlation between the irrelevant features and the category labels. This kind of spurious correlations is intrinsically caused by the unexpected correlations between irrelevant features and relevant features for a given category [18], [19], [20]. For graph-level tasks that we focus on in this paper, as the predicted properties of graphs are usually determined by the subgraph units (e.g., groups of atoms and chemical bonds representing functional units in a molecule) [1], [8], [21], we define one subgraph unit could be one relevant feature or irrelevant feature with graph label. Taking the classification task of "house" motif as an example, as depicted in Figure 1, where the graph is labeled by whether the graph has a "house" motif and the first column represents the testing graphs. The GCN

† *Corresponding author.*

- *S. Fan is with the Department of Computer Science, Beijing University of Posts and Telecommunications, Beijing 100876, China, and the Department of Computer Science and Technology in Tsinghua University, Beijing 100084, China. E-mail: fanshaohua@bupt.cn*
- *X. Wang is with the School of Software, Beihang University, Beijing, 100191, China. Email: xiao_wang@buaa.edu.cn*
- *C. Shi, and B. Wang are with the Department of Computer Science, Beijing University of Posts and Telecommunications, Beijing 100876, China. E-mail: {shichuan,wangbai}@bupt.edu.cn.*
- *P. Cui is with the Department of Computer Science and Technology in Tsinghua University, Beijing 100084, China. E-mail: cuip@tsinghua.edu.cn.*

Manuscript received Nov 17, 2021; revised May 1, 2023; revised Sep 15, 2023;

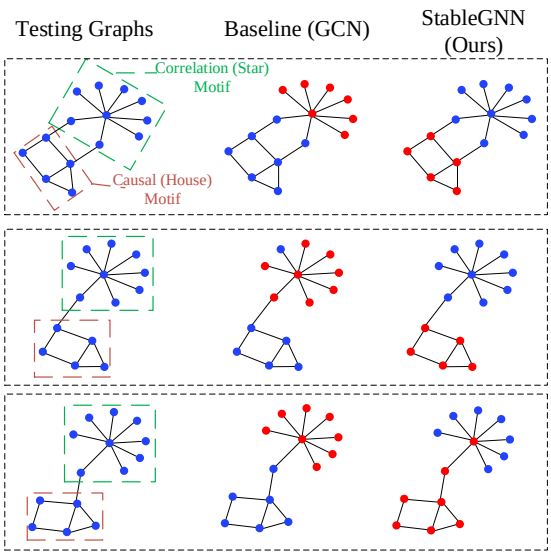


Fig. 1: Visualization of subgraph importance for "house" motif classification task, produced by the vanilla GCN model and StableGNN when most of training graphs containing "house" motifs with "star" motifs. The red subgraph indicates the most important subgraph used by the model for prediction (generated by GNNExplainer [1]). Due to the spurious correlation, the GCN model tends to focus more on "star" motifs while our model focuses mostly on "house" motifs. For more cases of testing graphs, please refer to Figure 5.

model is trained on the dataset where "house" motifs coexist with "star" motifs in most training graphs. With this dataset, the structural features of "house" motifs and "star" motifs would be strongly correlated. This unexpected correlation leads to spurious correlations between structural features of "star" motifs with the label "house". And the second column of Figure 1 shows the visualization of the most important subgraph used by the GCN for prediction (shown with red color and generated by GNNExplainer [1]). As a result, the GCN model tends to use such spurious correlation, i.e., "star" motif, for prediction. When encountering graphs without "star" motif, or other motifs (e.g., "diamond" motifs) with "star" motifs, the model is prone to make false predictions (See Section 5.1).

To improve the OOD generalization ability of GNNs, one important way is to make GNNs get rid of such subgraph-level spurious correlation. However, it is not a trivial task, which will face the two following challenges: (1) How to explicitly encode the subgraph-level information into graph representation? As the spurious correlation usually exists between subgraphs, it is necessary to encode such subgraph information into the graph representation, so that we can develop a decorrelation method based on the subgraph-level representation. (2) How to remove the spurious correlation among the subgraph-level representations? The nodes in the same type of subgraph units may exist correlation, for example, the information of 'N' atom and 'O' atom in 'NO_2' groups of molecular graphs may be encoded into different dimensions of learned embedding, but they act as an integrated whole and such correlations are stable across unseen testing distributions. Hence, we should not remove the correlation between the interior variables of one kind of subgraphs and only need to remove the spurious correlation between subgraph-level variables.

To address these two challenges, we propose a novel causal representation framework for graph, called **StableGNN**, which takes advantage of both the flexible representation ability of GNNs and the distinguishing ability for spurious correlations of causal inference methods. In terms of the first challenge, we propose a graph high-level variable learning module that employs a graph pooling layer to map nearby nodes to a set of clusters, where each cluster will be one densely-connected subgraph unit of original graph. Moreover, we theoretically prove that the semantic meanings of clusters would be aligned across graphs by an ordered concatenation operation. Given the aligned high-level variables, to overcome the second challenge, we analyze the degeneration of GNNs in causal view and propose a novel causal variable distinguishing regularizer to decorrelate each high-level variable pair by learning a set of sample weights. These two modules are jointly optimized in our framework. Furthermore, as shown in Figure 1, StableGNN can effectively partial out the irrelevant subgraphs (i.e., "star" motif) and leverage truly relevant subgraphs (i.e., "house" motif) for predictions.

In summary, the major contributions of the paper are as follows:

- To our best knowledge, we are one of the pioneer works studying the OOD generalization problem on GNNs for graph-level tasks, which is a key direction to apply GNNs to wild non-stationary environments.
- We propose a general causal representation learning framework for GNNs, jointly learning the high-level representation with the causal variable distinguisher, which could learn an invariant relationship between the graph causal variables with the labels. And our framework is general to be adopted for various base GNN models to help them get rid of spurious correlations.
- Comprehensive experiments are conducted on both synthetic datasets and real-world OOD graph datasets. The effectiveness, flexibility and interpretability of the proposed framework have been well-validated with convincing results.

2 Related Work

In this section, we discuss three main categories closely related to our work: graph neural networks, causal representation learning and stable learning.

2.1 Graph Neural Networks

Graph neural networks are powerful deep neural networks that could perform on graph data directly [2], [3], [4], [5], [7], [10]. GNNs have been applied to a wide variety of tasks, including node classification [3], [4], [5], [22], link prediction [23], [24], [25], graph clustering [26], [27], and graph classification [7], [10]. For graph classification, the task we majorly focus on here, the major challenge in applying GNNs is how to generate the representation of

the entire graph. Common approaches to this problem are simply summing up or averaging all the node embedding in the final layer. Several literatures argue that such simple operation will greatly ignore high-level structure that might be presented in the graph, and then propose to learn the hierarchical structure of graph in an end-to-end manner [8], [10], [28], [29]. Although these methods have achieved remarkable results in I.I.D setting, most of them largely ignore the generalization ability in OOD setting, which is crucial for real applications. During the review process, we notice several works claiming the importance of OOD generalization on graph classification tasks [30], [31], [32], [33]. Unlike the framework of OOD-GNN [30], which learns a single embedding for each graph and decorrelates each dimension of embeddings, our framework emphasizes the need to learn meaningful high-level representations for each subgraph and decorrelate those representations. DIR [31] divides a graph into causal- and non-causal part by an edge threshold. However, the threshold is set as the same for all graphs and is hard to select a good threshold for all graphs. CIGA [32] proposes to maximize the agreement between the invariant part of graphs with the same labels. For example, the functional groups NO_2 and NH_2 could both determine the mutagenicity of a molecule. However, subgraphs with the same labels may not always be identical. DisC [34] studies how to learn causal substructure in severe bias scenarios. Other methods [33], [35] are based on the environmental inference framework, in which these methods iteratively infer the environment labels of graphs and learn the invariant information based on the environment labels.

2.2 Causal Representation Learning

Recently, Schölkopf et al. [36] publish a survey paper, towards causal representation learning, and point out that causality, with its focus on representing structural knowledge about the data-generating process that allows interventions and changes, can contribute towards understanding and resolving some limitations of current machine learning methods. Traditional causal discovery and reasoning assume that the units are random variables connected by a causal graph. However, real-world observations are not structured into these units, for example, graph data we focus on is a kind of unstructured data. Thus for this kind of data, causal representation learning generally consists of two steps: (1) inferring the abstract/high-level causal variables from available low-level input features, and (2) leveraging the causal knowledge as an inductive bias to learn the causal structure of high-level variables. After learning the causal system, the causal relationship is robust to irrelevant interventions and changes in data distributions. Based on this idea, Invariant Risk Minimization (IRM) framework [37], [38], [39] proposes a regularization that enforces model learn an invariant representation across environments. The representation learning part plays the role of extracting high-level representation from low-level raw features, and the regularization encodes the causal knowledge that causal representation should be invariant across environments into the representation learning. Despite the IRM framework could learn the representations that have better OOD generalization ability, the requirement that domains should be labeled hinders the IRM methods from real applications. To overcome such a dilemma, some methods are proposed to implicitly learn domains from data [40], [41], [42], but they implicitly assume the training data is formed by balanced sampling from latent domains. However, all these methods are mainly designed for tabular or image data, thus they cannot capture the intrinsic properties of graphs.

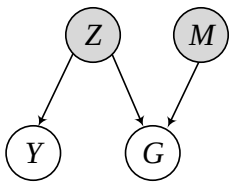


Fig. 2: Causal graph for data generation process. Gray nodes and white nodes mean the unobserved latent variables and the observed variables, respectively.

2.3 Stable Learning

To improve the feasibility of IRM-based methods, a series of researches on stable learning are proposed [43], [44]. These methods mainly bring the ideas from the causal effect estimation [45] into machine learning models. Particularly, Shen et al. [46] propose a global confounding balancing regularizer that helps the logistic regression model to identify causal features, whose causal effect on outcomes are stable across domains. To make the confounder balancing much easier in high-dimensional scenarios, Kuang et al. [43] utilize the autoencoder to encode the high-dimensional features into low-dimensional representation. Furthermore, [44] and [47] demonstrate that decorrelating relevant and irrelevant features can make a linear model produce stable predictions under distribution shifts. Nevertheless, they are all developed based on the linear framework. Recently, Zhang et al. [18] extend decorrelation into a deep CNN model to tackle more complicated data types like images. This method pays much attention to eliminating the non-linear dependence among features, but the feature learning part is largely ignored. We argue that it is important to develop an effective high-level representation learning method to extract variables with appropriate granularity for the targeted task, so that the causal variable distinguishing part could develop based on meaningful causal variables.

3 Problem Formulation

Notations. In this paper, for any vector $\mathbf{v} = (\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_n)$, let $\mathbf{v}_i$ the i-th element of $\mathbf{v}$. For any matrix $\mathbf{X}$, let $\mathbf{X}_{i,}$ and $\mathbf{X}_{,j}$ represent the i-th row and the j-th column in $\mathbf{X}$, respectively. $\mathbf{X}_{,j:k}$ denotes the submatrix of $\mathbf{X}$ from j-th column to $(k-1)$-th column. And for any italic uppercase letter, such as X, it will represent a random variable. We summarize the key notions used in the paper in Table 1.

Problem 1. ***OOD Generalization Problem on Graphs.*** *Given the training graphs $\mathcal{G}_{train} = \{(G_1, Y_1), \cdots, (G_n, Y_n)\}$, where G_i means the i-th graph data in the training set and Y_i is the*

TABLE 1: Glossary of notations.

Notations	Description
$\mathcal{G}_{train}/\mathcal{G}_{test}$	Training/Testing graphs
$\mathbf{A}$	Adjacency matrix
$\mathbf{F}$	Feature matrix
$\mathbf{S}$	Cluster assignment matrix
$\mathbf{Z}$	Node representation matrix
$\mathbf{H}$	High-level variable representation matrix
$\mathbf{P}$	Permutation matrix
T	Treatment variable
Y	Label/prediction variable
X	Confounder variable
$X^{(p)}$	The p-th high-level confounder variable
$\mathbf{X}^{(p)}$	The p-th high-level confounder matrix
$\mathbf{w}$	Sample/graph weights

corresponding label, the task is to learn a GNN model $h_\theta(\cdot)$ with parameter θ to precisely predict the label of testing graphs $\mathcal{G}_{test}$, where the distribution $\Psi(\mathcal{G}_{train}) \neq \Psi(\mathcal{G}_{test})$. And in the OOD setting, we do not know the distribution shifts from training graphs to unseen testing graphs.

We utilize a causal graph of the data generation process, as shown in Figure 2, to illustrate the fundamental reason to cause the distribution shifts on graphs. As illustrated in the figure, the observed graph data G is generated by the unobserved latent cause Z and M. Z is a set of relevant variables and the label Y is mainly determined by Z. M is a set of irrelevant variables which does not decisive for label Y. During the unseen testing process in the real world, the variable M could change, but the causal relationship $\mathrm{P}(Y \mid Z)$ is invariant across environments. Taking the classifying mutagenic property of a molecular graph [48] as an example, G is a molecular graph where the nodes are atoms and the edges are the chemical bonds between atoms, and Y is the class label, e.g., whether the molecule is mutagenic or not. The whole molecular graph G is an effect of relevant latent causes Z such as the factors to generate nitrogen dioxide (NO_2) group, which has a determinative effect on the mutagenicity of molecule, and the effect of irrelevant variable M, such as the carbon ring which exists more frequently in mutagenic molecules but not determinative [48]. If we aim to learn a GNN model that is robust to unseen change on M, such as carbon exists in the non-mutagenic molecule, one possible way is to develop a representation function $f(\cdot)$ to recover Z and M from G, and learn a classifier $g(\cdot)$ based on Z, so that the invariant causal relationship $P(Y|Z)$ could be learned.

4 The Proposed Method

4.1 Overview

The basic idea of our framework is to design a causal representation learning method that could extract meaningful graph high-level variables and estimate their true causal effects for graph-level tasks. As depicted in Figure 3, the proposed framework mainly consists of two components: the graph high-level variable learning component and causal variable distinguishing component. The graph high-level variable learning component first employs a graph pooling layer that learns the node embedding and maps nearby nodes into a set of clusters. Then we get the cluster-level embeddings through aggregating the node embeddings in the same cluster, and align the cluster semantic space across graphs through an ordered concatenation operation. The cluster-level embeddings act as high-level variables for graphs. After obtaining the high-level variables, we develop a sample reweighting component based on Hilbert-Schmidt Independence Criterion (HSIC) measure to learn a set of sample weights that could remove the non-linear dependencies among multiple multi-dimensional embeddings. As the learned sample weights could generate a pseudo-distribution with less spurious correlation among cluster-level variables, we utilize the weights to reweight the GNN loss. Thus the GNN model trained on this less biased pseudo-data could estimate the causal effect of each high-level variable on the label more precisely, resulting in better generalization ability in wild environments.

4.2 Graph High-level Variable Learning

As the goal of this component is to learn a representation that could represent the original subgraph unit explicitly, we adopt a differentiable graph pooling layer that could map densely-connected subgraphs into clusters in an end-to-end manner. And then we theoretically prove that the semantic meaning of the learned high-level representations could be aligned by a simple ordered concatenation operation.

High-level Variable Pooling. To learn node embedding as well as map densely connected subgraphs into clusters, we exploit the DiffPool layer [8] to achieve this goal by learning a cluster assignment matrix based on the learned embedding in each pooling layer. Particularly, as GNNs could smooth the node representations and make the representation more discriminative, given the input adjacency matrix $\mathbf{A}$, also denoted as $\mathbf{A}^{(0)}$ in this context, and the node features $\mathbf{F}$ of a graph, we firstly use an embedding GNN module to get the smoothed embeddings $\mathbf{F}^{(0)}$:

$$\mathbf{F}^{(0)} = \mathrm{GNN}^{(0)}_{\mathrm{embed}}(\mathbf{A}^{(0)}, \mathbf{F}), \tag{1}$$

where $\mathrm{GNN}^{(0)}_{\mathrm{embed}}(\cdot,\cdot)$ is a three-layer GNN module, and the GNN layer could be GCN [3], GraphSAGE [5], or others. Then we develop a pooling layer based on the smoothed representation $\mathbf{F}^{(0)}$. In particular, we first generate node representation at layer 1 as follows:

$$\mathbf{Z}^{(1)} = \mathrm{GNN}^{(1)}_{\mathrm{embed}}(\mathbf{A}^{(0)}, \mathbf{F}^{(0)}). \tag{2}$$

As nodes in the same subgraph would have similar node features and neighbor structure and GNN could map the nodes with similar features and structural information into similar representation, we also take the node embeddings $\mathbf{F}^{(0)}$ and adjacency matrix $\mathbf{A}^{(0)}$ into a pooling GNN module to generate a cluster assignment matrix at layer 1, i.e., the clusters of original graph:

$$\mathbf{S}^{(1)} = \mathrm{softmax}(\mathrm{GNN}^{(1)}_{\mathrm{pool}}(\mathbf{A}^{(0)}, \mathbf{F}^{(0)})), \tag{3}$$

where $\mathbf{S}^{(1)} \in \mathbb{R}^{n_0 \times n_1}$, and n_0 is the number of nodes of the input graph and n_1 is the number of the clusters at the layer 1, and $\mathbf{S}^{(1)}_{i,}$ represents the cluster assignment vector of i-th node, and $\mathbf{S}^{(1)}_{,j}$ corresponds to all nodes' assignment probabilities on j-th cluster at the layer 1. $\mathrm{GNN}^{(1)}_{\mathrm{pool}}(\cdot,\cdot)$ is

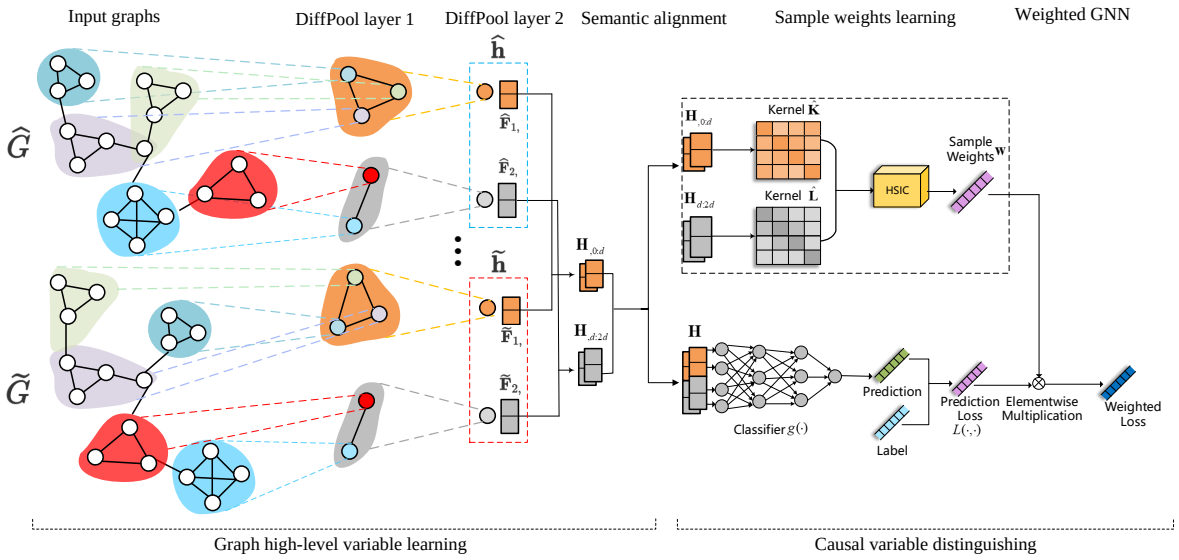


Fig. 3: The overall architecture of the proposed StableGNN.

also a three-layer GNN module, and the output dimension of $\text{GNN}_{\text{pool}}^{(1)}(\cdot,\cdot)$ is a pre-defined maximum number of clusters at layer 1 and is a hyperparameter. And the appropriate number of clusters could be learned in an end-to-end manner. The maximum number of clusters in the last pooling layer should be the number of high-level variables we aim to extract. The softmax function is applied in a row-wise fashion to generate the cluster assignment probabilities for each node at layer 1.

After obtaining the assignment matrix $\mathbf{S}^{(1)}$, we could know the assignment probability of each node on the predefined clusters. Hence, based on the assignment matrix $\mathbf{S}^{(1)}$ and the learned node embedding matrix $\mathbf{Z}^{(1)}$, we could get a new coarsened graph, where the nodes are the clusters learned by this layer and edges are the connectivity strength between each pair of clusters. Particularly, the new matrix of embeddings is calculated by the following equation:

$$\mathbf{F}^{(1)} = {\mathbf{S}^{(1)}}^{\mathrm{T}}\mathbf{Z}^{(1)} \in \mathbb{R}^{n_1 \times d}, \tag{4}$$

where d is the dimension of the embedding. This equation aggregates the node embedding $\mathbf{Z}^{(1)}$ according to the cluster assignment $\mathbf{S}^{(1)}$, generating embeddings for each of the n_1 clusters. Similarly, we generate a coarsened adjacency matrix as follows.

$$\mathbf{A}^{(1)} = {\mathbf{S}^{(1)}}^{\mathrm{T}}\mathbf{A}^{(0)}\mathbf{S}^{(1)} \in \mathbb{R}^{n_1 \times n_1}. \tag{5}$$

For all the operations with superscript (1), it is one DiffPool unit and it is denoted as $(\mathbf{A}^{(1)}, \mathbf{F}^{(1)}) = \text{DiffPool}(\mathbf{A}^{(0)}, \mathbf{F}^{(0)})$. For any DiffPool layer l, it could be denoted as $(\mathbf{A}^{(l)}, \mathbf{F}^{(l)}) = \text{DiffPool}(\mathbf{A}^{(l-1)}, \mathbf{F}^{(l-1)})$. In particular, we could stack multiple DiffPool layers to extract the deep hierarchical structure of the graph.

High-level Representation Alignment. After stacking L graph pooling layers, we could get the most high-level cluster embedding $\mathbf{F}^{(L)} \in \mathbb{R}^{n_L \times d}$, where $\mathbf{F}_{i,}^{(L)}$ represents the i-th high-level representation of the corresponding subgraph in the original graph and n_L is the number of high-level representation. As our target is to encode subgraph information into graph representation and $\mathbf{F}^{(L)}$ has explicitly encoded each densely-connected subgraph information into each row of $\mathbf{F}^{(L)}$, we propose to utilize the embedding matrix $\mathbf{F}^{(L)}$ to represent the high-level variables. However, due to the Non-Euclidean property of graph data, for the i-th learned high-level representations $\widehat{\mathbf{F}}_{i,}^{(L)}$ and $\widetilde{\mathbf{F}}_{i,}^{(L)}$ from two graphs $\widehat{G}$ and $\widetilde{G}$, respectively, their semantic meaning may not be matched, e.g., $\widehat{\mathbf{F}}_{i,}^{(L)}$ and $\widetilde{\mathbf{F}}_{i,}^{(L)}$ may represent scaffold substructure (e.g., carbon ring) and functional group (e.g., NO_2) in two molecular graphs, respectively. To match the semantic meaning of learned high-level representation across graphs, we propose to concatenate the high-level variables by the order of row index of high-level embedding matrix for each graph:

$$\mathbf{h} = \text{concat}(\mathbf{F}_{1,}^{(L)}, \mathbf{F}_{2,}^{(L)}, \cdots, \mathbf{F}_{n_L,}^{(L)}), \tag{6}$$

where $\mathbf{h} \in \mathbb{R}^{n_L d}$ and $\text{concat}(\cdot)$ means concatenation operation by the row axis. Moreover, we stack m high-level representations for a mini-batch with m graphs to obtain the embedding matrix $\mathbf{H} \in \mathbb{R}^{m \times n_L d}$:

$$\mathbf{H} = \text{stack}(\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_m), \tag{7}$$

where $\mathbf{h}_i$ is the concatenated high-level representation of sample i, $\text{stack}(\cdot)$ means the stacking operation by the row axis, and $\mathbf{H}_{i,}$ means the high-level representation for the i-th graph in the mini-batch. $\mathbf{H}_{i,(k-1)d:kd}$ means k-th high-level representation of i-th graph. Hence, to prove the semantic alignment of any two high-level representations $\mathbf{H}_{i,}$ and $\mathbf{H}_{j,}$, we need to demonstrate that the semantic meanings of $\mathbf{H}_{i,(k-1)d:kd}$ and $\mathbf{H}_{j,(k-1)d:kd}$ are aligned for all $k \in [1, n_L]$. To this end, we first prove the permutation invariant property of DiffPool layer.

Lemma 1. ***Permutation Invariance [8].*** *Given any permutation matrix* $\mathbf{P} \in \{0,1\}^{n \times n}$, *if* $\mathbf{P} \cdot GNN(\mathbf{A}, \mathbf{F}) = GNN(\mathbf{PAP}^{\mathrm{T}}, \mathbf{PF})$

(i.e., the GNN method used is permutation equivariant), then $\mathit{DiffPool}(\mathbf{A}, \mathbf{F}) = \mathit{DiffPool}(\mathbf{PAP}^{\mathrm{T}}, \mathbf{PF})$.

Proof. Following [49], a function $f : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times l}$: is *invariant* if $f(\mathbf{F}, \mathbf{X}) = f(\mathbf{PFP}^{\mathrm{T}}, \mathbf{PF})$, i.e, the permutation will not change node representation and the node order in the learned matrix. A function $f : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times l}$: is *equivariant* if $f(\mathbf{A}, \mathbf{F}) = \mathbf{P} \cdot f(\mathbf{PAP}^{\mathrm{T}}, \mathbf{PF})$, i.e., the permutation will not change the node representation but the order of nodes in matrix will be permuted. The permutation invariant aggregator functions such as mean pooling and max pooling ensure that the basic model can be trained and applied to arbitrarily ordered node neighborhood feature sets, i.e, permutation equivariant. Therefore, the Eq. (2) and (3) are the permutation equivariant by the assumption that the GNN module is permutation equivariant. And since any permutation matrix is orthogonal, i.e., $\mathbf{P}^{\mathrm{T}}\mathbf{P} = \mathbf{I}$, applying this into Eq. (4) and (5), we have:

$$\mathbf{F}^{(1)} = {\mathbf{S}^{(1)}}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{PZ}^{(1)}, \tag{8}$$

$$\mathbf{A}^{(1)} = {\mathbf{S}^{(1)}}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{PA}^{(1)}\mathbf{P}^{\mathrm{T}}\mathbf{PS}^{(1)}. \tag{9}$$

Hence, DiffPool layer is permutation invariant. □

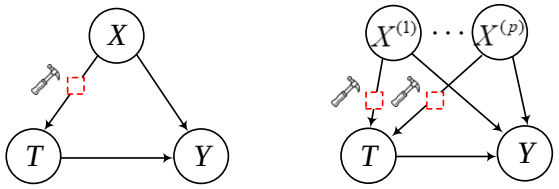
After proving the permutation invariant property of a single graph, we then illustrate the theoretical results for the semantic alignment of learned high-level representation across graphs.

Theorem 1. ***Semantic Alignment of High-level Variables.*** *Given any two graphs* $G_i = (\mathbf{A}_i, \mathbf{F}_i)$ *and* $G_j = (\mathbf{A}_j, \mathbf{F}_j)$*, the semantic space of high-level representations* $\mathbf{H}_i$, *and* $\mathbf{H}_j$, *learned by a series of shared DiffPool layers is aligned.*

Proof. WL-test [50] is widely used in graph isomorphism checking: if two graphs are isomorphic, they will have the same multiset of WL colors at any iteration. GNNs could be viewed as a continuous approximation to the WL test, where they both aggregate signal from neighbors and the trainable neural network aggregators is an analogy to the hash function in WL-test [10]. Therefore, the outputs of GNN module are exactly the continuous WL colors. The outputs of GNN layers or the WL color of nodes would represent their structural roles [10]. For example, the output assignment matrix of the pooling GNN module in Diffpool layer $\mathbf{S}^{(1)} \in \mathbb{R}^{n_0\times n_1}$ could be interpreted as the structural roles of n_0 input nodes with respect to the n_1 clusters of layer 1. Through the assignment matrix $\mathbf{S}^{(1)}$, it will aggregate the nodes with similar structural roles into the clusters with the same index. Hence, the semantic meaning of each column of the learned assignment matrices $\mathbf{S}_i^{(l)}$ and $\mathbf{S}_j^{(l)}$ of graphs G_i and G_j would be aligned, e.g., the k-th column would represent carbon ring structure in all molecule graphs, which act as the scaffold structural role in molecule graphs. Due to the permutation invariant property of DiffPool layer, according to Eq. (8), the input graph with any node order will map its carbon ring structure signal into the k-th high-level representation, i.e., the k-th high-level representation of $\mathbf{H}_{i,(k-1)d:kd}$ and $\mathbf{H}_{j,(k-1)d:kd}$. Hence, the semantic meaning of $\mathbf{H}_{i,.}$ and $\mathbf{H}_{j,.}$ is aligned. □

4.3 Causal Variable Distinguishing Regularizer

So far the variable learning part extracts all the high-level representations for all the densely-connected subgraphs regardless of whether it is relevant with the graph label due to causal relation or spurious correlation. In this section, we first analyze the reason leading to the degeneration of GNNs on OOD scenarios in a causal view and then propose the Causal Variable Distinguishing (CVD) regularizer with sample reweighting technique.



(a) Confounder balancing framework. (b) Multiple multi-dimensional confounder balancing framework.

Fig. 4: Causal view on GNNs.

Revisiting on GNNs in Causal View. As described in Section 3, our target is to learn a classifier $g(\cdot)$ based on the relevant variable Z. To this end, we need to distinguish which variable of learned high-level representation $\mathbf{H}$ belongs to Z or M. The major difference between Z and M is whether they have a causal effect on Y. For a graph-level classification task, after learning the graph representation, it will be fed into a classifier to predict its label. The prediction process could be represented by the causal graph with three variables and their relationships, as shown in Figure 4(a), where T is a treatment variable, Y is the prediction/outcome variable[1], and X is the confounder variable, which has effects both on the treatment variable and the outcome variable. The path $T \to Y$ represents the target of GNNs that aims to estimate the causal effect of the one learned variable T (e.g., i-th high-level representation $\mathbf{H}_{,(i-1)d:id}$) on Y. Meanwhile, other learned variables (e.g., j-th high-level representation $\mathbf{H}_{,(j-1)d:jd}$) will act as confounder X. Due to the existence of spurious correlations of subgraphs, there are spurious correlations between their learned embeddings. Hence, there incurs a path between X and T. [2] And because GNNs also employ the confounder (e.g., the representation of carbon ring) for prediction, there exists a path $X \to Y$. Hence, these two paths form a backdoor path between X and Y (i.e., $T \leftarrow X \to Y$) and induce the spurious correlation between T and Y. And the spurious correlation will amplify the true correlation between treatment variable and label, and may change in testing environments. Under this scenario, existing GNNs cannot estimate the causal effect of subgraphs accurately, so the performance of GNNs will degenerate when the spurious correlation change in the testing phase.

1. We use variable Y for both the ground-truth labels and prediction, as they are optimized to be the same.

2. Note that the direction of arrow means that the assignment of treatment value will dependent on the confounder. However, if the arrow is reversed, it will not affect the theoretical results in our paper.

Confounding balancing techniques [51], [52] correct the non-random assignment of treatment variable by balancing the distributions of confounder across different treatment levels. Because moments can uniquely determine a distribution, confounder balancing methods directly balance the confounder moments by adjusting weights of samples [51], [53]. The sample weights $\mathbf{w}$ for binary treatment scenario are learnt by:

$$\mathbf{w} = \arg\min_{\mathbf{w}} || \frac{\sum_{i:T_i=1} \mathbf{w}_i \cdot \mathbf{X}_{i,}}{\sum_{i:T_i=1} \mathbf{w}_i} - \frac{\sum_{j:T_j=0} \mathbf{w}_j \cdot \mathbf{X}_{j,}}{\sum_{j:T_j=0} \mathbf{w}_j} ||_2^2, \quad (10)$$

where $\mathbf{X}$ is the confounder matrix and $\mathbf{X}_{i,}$ is the confounder vector for i-th graph. Given a binary treatment feature T, $\frac{\sum_{i:T_i=1} \mathbf{w}_i \cdot \mathbf{X}_{i,}}{\sum_{i:T_i=1} \mathbf{w}_i}$ and $\frac{\sum_{j:T_j=0} \mathbf{w}_j \cdot \mathbf{X}_{j,}}{\sum_{j:T_j=0} \mathbf{w}_j}$ refer to the mean value of confounders on samples with and without treatment, respectively. After confounder balancing, the dependence between X and T (i.e., $T \leftarrow X$) would be eliminated, thus the correlation between the treatment variable and the output variable will represent the causal effect (i.e., $X \rightarrow Y$).

Moreover, for a GNN model, we have little prior knowledge on causal relationships between the learned high-level variables $\{\mathbf{H}_{,0:d}, \cdots, \mathbf{H}_{,(n_L-1)d:n_Ld}\}$, thus we have to set each learned high-level variable as treatment variable one by one, and the remaining high-level variables are viewed as confounding variables, e.g, $\mathbf{H}_{,0:d}$ is set as treatment variable and $\{\mathbf{H}_{,d:2d}, \cdots, \mathbf{H}_{,(n_L-1)d:n_Ld}\}$ are set as confounders. Note that for a particular treatment variable, previous confounder balancing techniques are mainly designed for a single-dimensional treatment feature as well as the confounder usually consists of multiple variables where each variable is a single-dimensional feature. In our scenario, however, as depicted in Figure 4(b), we should deal with the confounders which are composed of multiple multi-dimensional confounder variables $\{X^{(1)}, \cdots, X^{(p)}\}$, where each multi-dimensional confounder variable corresponds to one of learned high-level representations $\{\mathbf{H}_{,0:d}, \cdots, \mathbf{H}_{,(n_L-1)d:n_Ld}\}$. The multi-dimensional variable unit usually has integrated meaning such as representing one subgraph, so it is unnecessary to remove the correlation between the treatment variable with each of feature $\mathbf{H}_{,i}$ in one multi-dimensional feature unit, e.g., $\mathbf{H}_{,0:d}$. And we should only remove the subgraph-level correlation between treatment variable T with multiple multi-dimensional variable units $\{X^{(1)}, \cdots, X^{(p)}\}$. One possible way to achieve this goal is to learn a set of sample weights that balance the distributions of all confounding variables for the targeted treatment variable, as illustrated in Figure 4(b), i.e., randomizing the assignment of treatment T [3] with confounders $\{X^{(1)}, \cdots, X^{(p)}\}$. The sample weights $\mathbf{w}$ could be learnt by the following *multiple multi-dimensional confounder balancing* objective:

$$\mathbf{w} = \arg\min_{\mathbf{w}} \sum_{k=1}^{p} || \frac{\sum_{i:T_i=1} \mathbf{w}_i \cdot \mathbf{X}^{(k)}_{i,}}{\sum_{i:T_i=1} \mathbf{w}_i} - \frac{\sum_{j:T_j=0} \mathbf{w}_j \cdot \mathbf{X}^{(k)}_{j,}}{\sum_{j:T_j=0} \mathbf{w}_j} ||_2^2, \quad (11)$$

where $\mathbf{X}^{(k)}$ is the embedding matrix for k-th confounding variable $X^{(k)}$.

3. Here, we still assume the treatment is a binary variable. In the following part, we will consider the treatment as a multi-dimensional variable.

Weighted Hilbert-Schmidt Independence Criterion. Since the above confounder balancing method is mainly designed for binary treatment variable, which needs the treatment value to divide the samples into treated or control group, it is hard to apply to the continuous multi-dimensional variables learned by GNNs. Inspired by the intuition of confounder balancing which is to remove the dependence between the treatment with the corresponding confounding variables, we propose to remove dependence between continuous multi-dimensional random variable T with each of confounder in $\{X^{(1)}, \cdots, X^{(p)}\}$. Moreover, as the relationship between representations learned by the representation module is highly non-linear, it is necessary to measure the nonlinear dependence between them. And it is feasible to resort to HSIC measure [54]. For two random variables U and V and kernel k and l, HSIC is defined as $\text{HSIC}^{k,l}(U,V) := ||C^{k,l}_{UV}||^2_{\text{HS}}$, where $C^{k,l}$ is a cross-covariance operator in the Reproducing Kernel Hilbert Spaces (RKHS) of k and l [55], an RKHS analogue of covariance matrices. $||\cdot||_{\text{HS}}$ is the Hilbert-Schmidt norm, a Hilbert-space analogue of the Frobenius norm. For two random variables U and V and radial basis function (RBF) kernels k and l, $\text{HSIC}^{k,l}(U,V) = 0$ if and only if $U \perp V$. To estimate $\text{HSIC}^{k,l}(U,V)$ with finite sample, we employ a widely used estimator $\text{HSIC}^{k,l}_0(U,V)$ [55] with m samples, defined as:

$$\text{HSIC}^{k,l}_0(U,V) = (m-1)^{-2} tr(\mathbf{KPLP}), \quad (12)$$

where $\mathbf{K}, \mathbf{L} \in \mathbb{R}^{m \times m}$ are RBF kernel matrices containing entities $\mathbf{K}_{ij} = k(U_i, U_j)$ and $\mathbf{L}_{ij} = l(V_i, V_j)$. $\mathbf{P} = \mathbf{I} - m^{-1}\mathbf{1}\mathbf{1}^{\mathrm{T}} \in \mathbb{R}^{m \times m}$ is a centering matrix, where $\mathbf{I}$ is an identity matrix and $\mathbf{1}$ is an all-one column vector. $\mathbf{P}$ is used to center the RBF kernel matrices to have zero mean in the feature space.

To eliminate the dependence between the high-level treatment representation with the corresponding confounders, sample reweighting techniques could generate a pseudo-distribution that has less dependence between variables [18], [56]. We propose a sample reweighting method to eliminate the dependence between high-level variables, where the non-linear dependence is measured by HSIC.

We use $\mathbf{w} \in \mathbb{R}^m_+$ to denote a set of sample weights. For any two random variables U and V, we first utilize the random initialized weights to reweight these two variables:

$$\hat{U} = (\mathbf{w} \cdot \mathbf{1}^{\mathrm{T}}) \odot U, \quad (13)$$

$$\hat{V} = (\mathbf{w} \cdot \mathbf{1}^{\mathrm{T}}) \odot V, \quad (14)$$

where '$\odot$' is the Hadamard product. Substituting $\hat{U}$ and $\hat{V}$ into Eq. (12), we obtain the weighted HSIC value:

$$\hat{\text{HSIC}}^{k,l}_0(U,V,\mathbf{w}) = (m-1)^{-2} tr(\hat{\mathbf{K}}\mathbf{P}\hat{\mathbf{L}}\mathbf{P}), \quad (15)$$

where $\hat{\mathbf{K}}, \hat{\mathbf{L}} \in \mathbb{R}^{m \times m}$ are weighted RBF kernel matrices containing entries $\hat{\mathbf{K}}_{ij} = k(\hat{U}_i, \hat{U}_j)$ and $\hat{\mathbf{L}}_{ij} = l(\hat{V}_i, \hat{V}_j)$. Specifically, for treatment variable $\mathbf{H}_{,0:d}$ and its corresponding multiple confounder variables $\{\mathbf{H}_{,d:2d}, \cdots, \mathbf{H}_{,(n_L-1)d:n_Ld}\}$, we share the sample weights $\mathbf{w}$ across multiple confounders and propose to optimize $\mathbf{w}$ by:

$$\mathbf{w}^* = \arg\min_{\mathbf{w} \in \Delta_m} \sum_{1<p<n_L} \hat{\text{HSIC}}^{k,l}_0(\mathbf{H}_{,0:d}, \mathbf{H}_{,(p-1)d:pd}, \mathbf{w}), \quad (16)$$

where $\Delta_m = \{\mathbf{w} \in \mathbb{R}_+^m | \sum_{i=1}^m \mathbf{w}_i = m\}$ is used to control the overall loss of each batch almost unchange, and we utilize $\mathbf{w} = \text{softmax}(\mathbf{w})$ to satisfy this constraint. Hence, reweighting training samples with the optimal $\mathbf{w}^*$ can mitigate the dependence between high-level treatment variable with confounders to the greatest extent.

Global Multi-dimensional Variable Decorrelation. Note that the above method is to remove the correlation between a single treatment variable $\mathbf{H}_{,0:d}$ with the confounders $\{\mathbf{H}_{,d:2d}, \cdots, \mathbf{H}_{,(n_L-1)d:n_Ld}\}$. However, we need to estimate the causal effect of all the learned high-level representations $\{\mathbf{H}_{,0:d}, \mathbf{H}_{,d:2d}, \cdots, \mathbf{H}_{,(n_L-1)d:n_Ld}\}$. As mentioned above, we need to set each high-level representation as a treatment variable and the remaining high-level representations as confounders, and remove the dependence between each treatment variable with the corresponding confounders. One effective way to achieve this goal is to remove all the dependence between variables. Specifically, we learn a set of sample weights that globally remove the dependence between each pair of high-level representations, defined as follows:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}\in\Delta_m} \sum_{1\le i<j\le n_L} \hat{\text{HSIC}}_0^{k,l}(\mathbf{H}_{,(i-1)d:id}, \mathbf{H}_{,(j-1)d:jd}, \mathbf{w}). \tag{17}$$

As we can see from Eq. (17), the global sample weights $\mathbf{w}$ simultaneously reduce the dependence among all high-level representations.

4.4 Weighted Graph Neural Networks

In the traditional GNN model, the parameters of the model are learned on the original graph dataset $\mathcal{G} = \{G_1, \cdots, G_m\}$. Because the sample weights $\mathbf{w}$ learned by the causal variable distinguishing regularizer are capable of globally decorrelating the high-level variables, we propose to use the sample weights to reweight the GNN loss, and iteratively optimize sample weights $\mathbf{w}$ and the parameters of weighted GNN model as follows:

$$f^{(t+1)}, g^{(t+1)} = \arg\min_{f,g} \sum_{i=1}^m \mathbf{w}_i^{(t)} L(g(f(\mathbf{A}_i, \mathbf{F}_i)), y_i), \tag{18}$$

$$\mathbf{w}^{(t+1)} = \arg\min_{\mathbf{w}^{(t+1)}\in\Delta_m} \sum_{1\le i<j\le n_L} \hat{\text{HSIC}}_0^{k,l}(\mathbf{H}^{(t+1)}_{,(i-1)d:id}, \mathbf{H}^{(t+1)}_{,(j-1)d:jd}, \mathbf{w}^{(t)}), \tag{19}$$

where $f(\cdot)$ is the representation part of our model and its output is the high-level representation $\mathbf{H}$, $\mathbf{H}^{(t+1)} = f^{(t+1)}(\mathbf{A}, \mathbf{F})$, t represents the iteration number, $g(\cdot)$ is a linear prediction layer, and $L(\cdot,\cdot)$ represents the loss function depends on which task we target. When updating the sample weights and the parameters of GNN model being fixed, we need to optimize the objective function Eq. (19). We update the sample weights by mature optimization technique, i.e., Adam. After obtaining the sample weight of each graph in the batch, we optimize the objective function of weighted GNNs Eq. (18) by an Adam optimizer [57]. For the classification task, cross-entropy loss is used, and for the regression task, least squared loss is used. Initially, $\mathbf{w}^{(0)} = (1, 1, \cdots, 1)^{\mathrm{T}}$ in each mini-batch. In the training phase, we iteratively optimize sample weights and model parameters with Eq. (18) and (19). During the inference phase, the predictive model directly conducts prediction based on the GNN model without any calculation of sample weights. The detailed procedure of our model is shown in Algorithm 1.

Although StableGNN still performs on dataset $\mathcal{G}$, the weight $\mathbf{w}_i$ of each graph is no longer same. This weight adjusts the contribution of each graph in the mini-batch loss, so that the GNN parameters are learned on the dataset that each high-level features are decorrelated[4] which can better learn the true correlation between relevant features and labels.

Algorithm 1: Training process of StableGNN

Input: Training graphs $\mathcal{G}_{train} = \{(G_1, y_1), \cdots, (G_N, y_N)\}$;
Training Epoch:$Epoch$;
Decorrelation Epoch: $DecorEpoch$;
Output: Learned GNN model;

```
while t < Epoch do
    for 1 to BatchNumber do
        Forward propagation to generate H;
        for 1 to DecorEpoch do
            Optimize sample weights w via Eq. (19);
        end
        Back propagate with weighted GNN loss to
         update f and g via Eq. (18);
    end
end
```

Discussions. Our proposed framework, StableGNN, aims to relieve the distribution shifts problem by causal representation learning diagram in a general way. Specifically, a new causal representation learning for graphs that seamless integrates the power of representation learning and causal inference is proposed. For the representation learning part, we could utilize various state-of-the-art graph pooling layer [8], [10], [28], [29] to extract high-level representations, nevertheless, the main intuition of our work is that we should learn high-level meaningful representation rather than meaningless mixed representation in our proposed learning diagram. This point is key for meaningful and effective causal learning, which is validated in Section 5.2.

Limitations. In our model, we assume a general causal variable relationships in Figure 4. Nevertheless, for some datasets or tasks, there may exist more complicated causal relationships among high-level variables, hence discovering causal structure for these high-level variables may be useful for reconstructing the latent data generation process and achieving better generalization ability.

Time Complexity Analysis. As the proposed framework consists of two parts, we analyze them separately. For the graph high-level representation learning part, to cluster nodes, although it requires the additional computation of an assignment matrix, we observed that the Diffpool layer did not incur substantial additional running time in practice. The reason is that each DiffPool layer reduces the size of graphs

4. In this paper, we slightly abuse the term "decorrelate/decorrelation", which means removing both the linear and non-linear dependence among features unless specified.

by extracting a coarser high-level representation of the graph, which speeds up the graph convolution operation in the next layer. For CVD regularizer, given n_L learned high-level representation, the complexity of computing HSIC value of each pair of high-level variables is $\mathcal{O}(m^2)$ [58], where m is the batch size. Hence, for each batch, the computation complexity of CVD regularizer is $\mathcal{O}(tn_L(n_L-1)m^2)$, where t is the number of epochs to optimize $\mathbf{w}$ and n_L is a very small number.

5 Experiments

In this section, we evaluate our algorithm on both synthetic and real-world datasets, comparing with state-of-the-arts.

Model Configurations. In our experiments, the GNN layer used for DiffPool layer is built on top of the GraphSAGE [5] or GCN [3] layer, to demonstrate that our framework can be applied on top of different GNN models. We use the "max pooling" variant of GraphSAGE. One DiffPool layer is used for all the datasets and more sophisticated hierarchical layers could be learned by stacking multiple DiffPool layers. For our model, StableSAGE/StableGCN refers to using GraphSAGE/GCN as base model, respectively. All the models are trained with same learning rate mode and the model for prediction is selected based on the epoch with best validation performance.

5.1 Experiments on Synthetic Data

To better verify the advantages of StableGNN on datasets with different degrees of distribution shifts between training set and testing set, we generate the synthetic datasets with a clear generation process so that the bias degree of datasets is controllable.

Dataset. We aim to generate a graph classification dataset that has a variety of distribution shifts from training dataset to testing dataset. Inspired by recent studies on GNN explanation [1], [59], we utilize motif as a subgraph of graphs. We first generate a base subgraph for each graph, that is, each positive graph has a "house"-structured network motif and each negative graph has a motif that is randomly drawn from 4 candidate motifs (i.e., star, clique, diamond and grid motifs). Hence, the "house" motif is the causal structure that causally determines the label. To inject spurious correlation, $\mu * 100\%$ of positive graphs will be added "star" motif and the remaining positive and negative graphs will randomly add a non-causal motif from 4 candidate motifs. The node features are drawn from the same uniform distribution for all nodes. We set μ as {0.6, 0.7, 0.8, 0.9} to get four spurious correlation degrees for the training set. And we set $\mu = 0.5$ to generate OOD validation set and $\mu = 0.25$ to generate an unbiased testing dataset. The larger μ for the training set means there incurs a larger distribution shift between the training and testing sets. The resulting graphs are further perturbed by adding edges from the vertices of the first motif to one of the vertices of the second motif in each graph with the probability 0.25. The number of training samples is 2000, and for validation and testing set is 1000.

Experimental Settings and Metrics. As the synthetic data has a known generating mechanism and our model is based on the GraphSAGE/GCN, to clearly validate and explain the effectiveness of our framework helping base GNN get rid of spurious correlation, in this subsection, we only compare with the base models. The number of layers of GraphSAGE and GCN is set as 5. The dropout rate for all the methods is set as 0.0. For all the GNN models, an initial learning rate is set as 1×10^{-3}, the reduce factor is 0.5, and the patience value is 5. And the learning rate of CVD regularizer is selected from $\{0.1, 0.3, \cdots, 1.3\}$. To baselines, the training epoch is set as 50. For Stable-SAGE/GCN, we set 20 epochs to warm up the parameters, i.e., training without the CVD regularizer, and 30 epochs to train the whole model. The decorrelation epoch to learn sample weights is set as 50. For all the methods, we take the model of the epoch with the best validation performance for prediction. The batch size is set as 250. The maximum number of clusters (i.e., high-level representations) is set as 7 for StableSAGE and 8 for StableGCN. For all the baseline models, if not mentioned specially, we aggregate node embeddings by mean pooling readout function to get the graph-level representation. Following [15], we augment each GNN layer with batch normalization (BN) [60] and residual connection [61]. We evaluate the methods with three widely used metrics for binary classification, i.e., Accuracy, F1 score and ROC-AUC [62]. For all the experiments, we run 4 times with different random seeds and report their mean and standard error of prediction value with the corresponding metric on the test set in percent.

Results on Synthetic Data. The results are given in Table 2, and we have the following observations. First, both the GraphSAGE and GCN suffer from serious performance decrease with the increase of spurious correlation degree, e.g., for F1 score, GraphSAGE drops from 67.83 to 54.24, and GCN drops from 67.06 to 62.28, indicating that spurious correlation greatly affects the GNNs' generalization performance and the heavier distribution shifts will cause a larger performance decrease. Second, compared with the base model, our proposed models achieve up to 19.64% performance improvements, and gain larger improvements under heavier distribution shifts. As we know the "house" motif is decisive for the label, the only way to improve the performance is to utilize this causal subgraph, demonstrating that our models could significantly reduce the influence of spurious correlation among subgraphs and reveal the true relationship between causal subgraphs with labels. Third, when building our framework both on GraphSAGE and GCN, our framework could achieve consistent improvements, and it indicates that StableGNN is a general framework and has the potential to adapt to various GNN architectures.

Explanation Analysis. An intuitive type of explanation for GNN models is to identify subgraphs that have a strong influence on final decisions [1]. To demonstrate whether the model focuses on the relevant or irrelevant subgraphs while conducting prediction, we utilize GNNExplainer [1] to calculate the most important subgraph with respect to GNN's prediction and visualize it with red color. As GNNExplainer needs to compute an edge mask for explanation and GraphSAGE cannot be aware of the edge weights, we explain the GCN-based model, i.e., GCN and StableGCN. As shown in Figure 5, we find the following interesting cases

TABLE 2: Results on synthetic datasets in different settings. The 'improvements' means the improvement percent of StableSAGE/StableGCN against GraphSAGE/GCN.

Correlation Degree (μ)	Method	Accuracy	F1 score	ROC-AUC
0.6	GraphSAGE	69.68±0.91	67.83±1.41	77.49±0.71
	StableSAGE	73.93±0.66	73.05±1.21	79.22±1.07
	Improvements	6.10%	7.70%	2.23%
0.7	GraphSAGE	67.85±0.76	63.76±1.32	75.55±1.01
	StableSAGE	73.9±1.78	71.23±1.88	81.48±4.13
	Improvements	8.92%	11.71%	7.85%
0.8	GraphSAGE	65.67±1.22	60.23±0.81	72.70±1.45
	StableSAGE	72.15±1.26	68.56±0.87	81.35±2.12
	Improvements	9.86%	13.83%	9.98%
0.9	GraphSAGE	65.2±0.94	54.24±1.98	72.89±0.67
	StableSAGE	70.35±1.66	64.84±2.54	80.31±1.78
	Improvements	7.90%	19.54%	10.18%
0.6	GCN	70.98±0.93	67.06±2.95	77.55±0.55
	StableGCN	74.92±1.91	73.91±2.49	81.79±2.42
	Improvements	5.56%	10.21%	5.47%
0.7	GCN	70.9±1.45	65.57±3.69	78.27±1.53
	StableGCN	73.15±2.62	70.29±2.76	79.77±3.42
	Improvements	3.17%	7.198%	1.92%
0.8	GCN	70.35±0.50	65.41±0.85	75.76±1.09
	StableGCN	74.5±1.03	70.94±1.69	81.53±0.86
	Improvements	5.90%	8.45%	7.62%
0.9	GCN	69.68±0.56	62.28±1.38	76.61±0.66
	StableGCN	76.35±1.37	72.64±2.62	83.24±0.58
	Improvements	9.57%	16.63%	8.65%

contributing to the success of our model, where each case represents a kind of failure easily made by existing methods.

- **Case 1.** As we can see, GCN assigns the higher weights to "star" motif, however, StableGCN concentrates more on "house" motif. Although GCN could make correct predictions based on the "star" motif, which is highly correlated with "house" motif, this prediction is unstable and it means that if there incurs spurious correlations in the training set, the model could learn the spurious correlation and rely on this clue for predictions. This unstable prediction is undesirable, as the unstable structure may change during the wild testing environments.
- **Case 2.** In this case, there is a "grid" motif connecting with "house" motif. As we can see, GCN pays more attention on "grid" motif and StableGCN still concentrates on "house" motif. Due to the existence of spurious correlated subgraphs, it will reduce the confidence of the true causal subgraph for prediction. When there appears another irrelevant subgraph, GCN may also pay attention on this irrelevant subgraph, leading to incorrect prediction results. However, our model could focus on the true causal subgraphs regardless of which kind of subgraphs are associated with them.
- **Case 3.** This is a case for negative samples. The spurious correlation leads to the GCN model focusing on the "star" motif. As the "star" motif is correlated with the positive label, GCN model will predict this graph as a positive graph. In contrast, due to the decorrelation of subgraphs in our model, we find that the "star" motif may not discriminate to the label decision and "diamond" motif may attribute more to the negative labels.

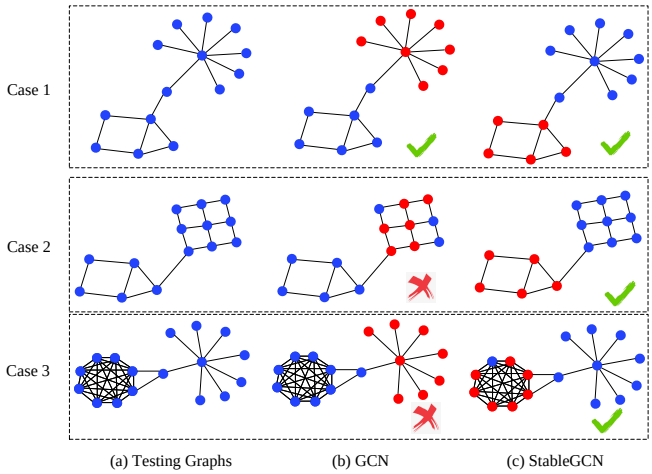


Fig. 5: Explanation cases of GCN and StableGCN. Red nodes are the important subgraph calculated by the GNNExplainer.

5.2 Experiments on Real-world Datasets

In this section, we apply our StableGNN algorithm on eight real-world datasets for out-of-distribution graph property prediction.

Datasets. We adopt seven molecular property prediction datasets from OGB Datasets [6]. All the molecules are pre-processed using RDKit [63]. Each graph represents a molecule, where nodes are atoms, and edges are chemical bonds. We use the 9-dimensional node features and 3-dimensional edges features provided by [6], which has better generalization performance. The task of these datasets is to predict the target molecular properties, e.g., whether a molecule inhibits HIV virus replication or not. Input edge features are 3-dimensional, containing bond type, bond stereochemistry as well as an additional bond feature indicating whether the bond is conjugated. Depending on the properties of molecular, these datasets can be categorized into three kinds of tasks: binary classification, multi-label classification and regression. Different from commonly used random splitting, these datasets adopt a scaffold splitting [64]

procedure that splits the molecules with different scaffolds into training or testing sets. All the molecules with the same scaffold could be treated as an environment, and the scaffold splitting attempts to separate molecules with different scaffolds into different subsets. For example, two molecules, Cyclopropanol (C3H6O) and 1,4-Cyclohexanediol (C6H12O2), contain different scaffold patterns: the former scaffold is 3C-ring and the latter is 6C-ring. Although sampled with different distributions, they are both readily soluble in water due to the invariant subgraph hydroxy (-OH) attached to different scaffolds [65]. The environments of training and testing sets are different, resulting in different graph distributions. Therefore, this kind of data could be used to test whether the model could leverage the causal subgraph to make predictions, i.e., the generalization ability of GNNs on graphs with different distributions. Moreover, we also conduct the experiments on a commonly used graph classification dataset, MUTAG [66], as we could explain the results based on the knowledge used in [48]. It consists of 4,337 molecule graphs. Each graph is assigned to one of 2 classes based on its mutagenic effect. Note that this dataset cannot adopt the scaffold splitting, as the +4 valence N atom, which commonly exists in this dataset, is illegal in the RDKit [67] tool used for scaffold splitting. We just use the random splitting for this dataset, however, we still believe there are some OOD cases in the testing set. The splitting ratio for all the datasets is 80/10/10. The detailed statistics are shown in Table 3.

Baselines. As the superiority of GNNs against traditional methods on graph-level tasks, like kernel-based methods [68], [69], has been demonstrated by previous literature [8], here, we mainly consider baselines based upon several related and state-of-the-art GNNs.

- Base models: GraphSAGE [5] and GCN [3] are classical GNN methods. We utilize them as base models in our framework, so they are the most related baselines to validate the effectiveness of the proposed framework.
- DIFFPOOL [8]: It is a hierarchical graph pooling method in an end-to-end fashion. It adopts the same model architecture and hyperparamter setting with high-level variable learning components in our framework, except that DIFFPOOL model aggregates the clusters' representation by summation operation rather than an ordered concatenation operation to generate the final high-level representations.
- GAT [4]: It is an attention-based GNN method, which learns the edge weights by an attention mechanism.
- GIN [70]: It is a graph isomorphism network that is as powerful as the WL graph isomorphism test. We compare with two variants of GIN, i.e., whether the ϵ is a learnable parameter or a fixed 0 scalar, denoted as GIN and GIN0, respectively.
- MoNet [71]: It is a general architecture to learn on graphs and manifolds using the bayesian gaussian mixture model.
- SGC [72]: It is a simplified GCN-based method, which reduces the excess complexity through successively removing nonlinearities and collapsing weight matrices between consecutive layers.
- JKNet [73]: It is a GNN framework that selectively combines different aggregations at the last layer.
- DIR [31]: It is a GNN method designed for the distribution shift problem, which disentangles the casual and non-causal subgraphs.
- CIGA [32]: It is a GNN method that learns causally invariant representations for OOD generalization on graphs.

Experimental Settings and Metrics. Here, we only describe the experimental settings that are different from Section 5.1. For all GNN baselines, we follow [6] which set the number of layers as 5. The dropout rate after each layer for all the methods on the datasets for binary classification and multi-label classification is set as 0.5, and for regression datasets, the dropout rate is set as 0.0. The number of batch size for binary classification and multi-label classification is set as 128, and for regression datasets, the batch size is set as 64. The training epoch for all the baselines is set as 200. And for our model, we utilize 100 epochs to warm up and 100 epochs to train the whole model. The maximum number of clusters is set as 7 for our models and DIFFPOOL. For DIR and CIGA, the hyperparameter s_c the selection ratio of casual subgraph is chosen from {0.1, 0.2, 0.25, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}, and the hyperparameters α and β for contrastive loss and hinge loss of CIGA are both chosen from {0.5, 1, 2, 4, 8, 16, 32} according to the validation performances. Moreover, as the loss of DIR and CIGA are specifically designed for the multi-classification task, they cannot perform on the datasets with multi-label classification and regression tasks. As these datasets usually treat chemical bond type as their edge type, to include edge features, we follow [6] and add transformed edge features into the incoming node features. For the datasets from OGB, we use the metrics recommended by the original paper for each task [6]. And following [1], we adopt Accuracy for MUTAG dataset.

Results on Real-world Datasets. The experimental results on eight datasets are presented in Table 4, and we have the following observations. First, comparing with these competitive GNN models, Stable-SAGE/GCN achieves 6 rank one and 2 rank two on all eight datasets. And the average rank of StableSAGE and StableGCN are 1.75 and 2.87, respectively, which is much higher than the third place, i.e., 4.38 for MoNet. It means that most existing GNN models cannot perform well on OOD datasets and our model significantly outperforms existing methods, which well demonstrates the effectiveness of the proposed causal representation learning framework. Second, compared with the base models, i.e., GraphSAGE and GCN, our models achieve consistent improvements on all datasets, validating that our framework could boost the existing GNN architectures. Third, Stable-SAGE/GCN also outperforms DIFFPOOL method by a large margin. Although we utilize the DiffPool layer to extract high-level representations in our framework, the seamless integration of representation learning and causal learning is the key to the improvements of our model. Fourth, Stable-SAGE/GCN outperforms DIR and CIGA, showing the effectiveness of our model over them on OOD generalization problem on graphs. And DIR and CIGA need to set a hyperparameter for the selection ratio of casual subgraph for all graphs, where the selection ratio is fixed and the same for all graphs and it

TABLE 3: Summary of real-world datasets.

Dataset	Task Type	#Graphs	Average #Nodes	Average #Edges	#Task	Splitting Type	Metric
Molbace	Binary classification	1,513	34.1	36.9	1	Scaffold splitting	ROC-AUC
Molbbbp	Binary classification	2,039	24.1	26.0	1	Scaffold splitting	ROC-AUC
Molhiv	Binary classification	41,127	25.5	27.5	1	Scaffold splitting	ROC-AUC
MUTAG	Binary classification	4,337	30.32	30.77	1	Random splitting	Accuracy
Molclintox	Multi-label classification	1,477	26.2	27.9	2	Scaffold splitting	ROC-AUC
Moltox21	Multi-label classification	7,831	18.6	19.3	12	Scaffold splitting	ROC-AUC
Molesol	Regression	1,128	13.3	13.7	1	Scaffold splitting	RMSE
Mollipo	Regression	4,200	27.0	29.5	1	Scaffold splitting	RMSE

TABLE 4: Performance of real-world graph datasets. The number in the $(\cdot)$ along with each performance number means the rank of the method among all the methods on this dataset. Because the losses of DIR and CIGA are designed for binary/multi-classifcation task, they cannot perform on the datasets with multilabel classification and regression tasks. Hence, we only show the results of them on binary classification task and do not rank them. "↑" means that for this metric, the larger value means better performance. "↓" means that for this metric, the smaller value means better performance. Best results of all methods are indicated in bold.

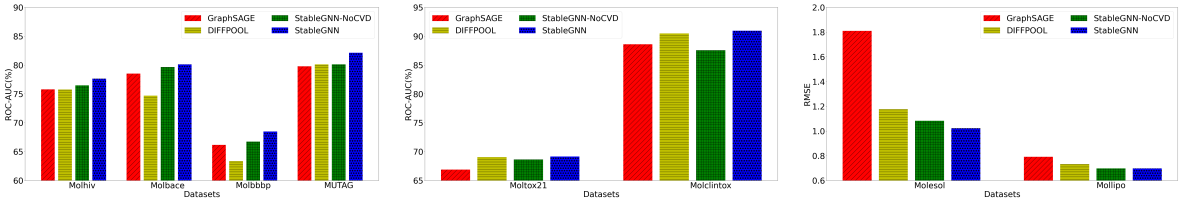
Methods	Binary Classification			
	Molhiv ROC-AUC (↑)	Molbace ROC-AUC (↑)	Molbbbp ROC-AUC (↑)	MUTAG Accuracy (↑)
GIN	76.21±0.53 (6)	74.50±2.75 (9)	67.72±1.89 (4)	78.86±1.15 (10)
GIN0	75.49±0.91 (9)	74.36±3.48 (10)	66.65±1.32 (6)	79.03±0.56 (9)
GAT	76.52±0.69 (5)	**81.17±0.8 (1)**	67.17±0.49 (5)	79.26±1.13 (8)
MoNet	77.13±0.79 (3)	76.92±0.91 (6)	**69.52±0.46 (1)**	79.95±0.86 (4)
SGC	69.46±1.44 (11)	71.28±1.79 (11)	61.17±2.91 (11)	69.53±0.77 (11)
JKNet	74.99±1.60 (10)	78.99±13.4 (3)	65.62±0.77 (9)	79.49±1.16 (7)
DIFFPOOL	75.75±1.38 (8)	74.69±11.13 (8)	63.35±2.21 (10)	80.13±1.32 (3)
DIR	61.40±10.69	60.81±10.47	54.65±8.08	62.61±3.00
CIGA	60.24±9.13	72.52±8.43	60.06±5.66	74.12±6.19
GCN	76.63±1.13 (4)	75.88±1.85 (7)	66.47±0.90 (7)	79.89±1.32 (5)
StableGCN	**77.79 ±1.19 (1)**	76.95±3.27 (5)	68.82±3.87 (2)	81.45±2.00 (2)
GraphSAGE	75.78±2.19 (6)	78.51±1.72 (4)	66.16±0.97 (8)	79.78±0.78 (6)
StableSAGE	77.63±0.79 (2)	80.73±3.98 (2)	68.47±2.47 (3)	**82.13±0.32 (1)**

Methods	Multilabel Classification		Regression		
	Molclintox ROC-AUC (↑)	Moltox21 ROC-AUC (↑)	Molesol RMSE (↓)	Molipo RMSE (↓)	Average-Rank (↓)
GIN	86.86±3.78 (6)	64.20±0.23 (11)	1.1002±0.0450 (4)	0.8051±0.0323 (10)	7.50 (9)
GIN0	89.31±2.11 (3)	64.62±0.87 (10)	1.1358±0.0587 (5)	0.8050±0.0123 (9)	7.63 (10)
GAT	83.47±1.37 (8)	68.81±0.48 (5)	1.2758±0.0269 (10)	0.8101±0.0183 (9)	6.38 (7)
MoNet	86.75±1.22 (7)	67.02±0.26 (7)	1.0753±0.0357 (3)	0.7379±0.0117 (4)	4.38 (3)
SGC	77.76±1.87 (10)	66.49±1.10 (8)	1.6548±0.0462 (11)	1.0681±0.0148 (10)	10.38 (11)
JKNet	81.63±2.79 (9)	65.98±0.46 (9)	1.1688±0.0434 (7)	0.7493±0.0048 (5)	7.38 (8)
DIFFPOOL	90.48±2.42 (2)	69.05±0.94 (3)	1.176±0.01388 (8)	0.7325±0.0221 (2)	5.50 (4)
GCN	86.23±2.81 (7)	67.75±0.66 (6)	1.153±0.0392 (6)	0.7927±0.0086 (8)	6.25 (6)
StableGCN	87.98±2.37 (5)	**70.80±0.31 (1)**	**0.9638±0.0292 (1)**	0.7839±0.0165 (6)	2.87 (2)
GraphSAGE	88.60±2.44 (4)	68.88±0.59 (4)	1.1852±0.0353 (9)	0.7911±0.0147 (7)	6.00 (5)
StableSAGE	**90.96±1.93 (1)**	69.14±0.24 (2)	1.0092±0.0706 (2)	**0.6971±0.0297 (1)**	**1.75 (1)**

is hard to set in practice. Fifth, Stable-SAGE/GCN achieves superior performance on datasets with three different tasks and a wide range of dataset scales, indicating that our proposed framework is general enough to be applied to datasets with a variety of properties.

Ablation Study. Note that our framework naturally incorporates high-level representation learning and causal effect estimation in a unified framework. Here we conduct ablation studies to investigate the effect of each component. For our framework without the CVD regularizer, we term it as StableGNN-NoCVD. [5] The results are presented in Figure 6. We first find that StableGNN-NoCVD outperforms GraphSAGE on most datasets, indicating that learning hierarchical structure by our model for graph-level tasks is necessary and effective. Second, StableGNN-NoCVD achieves competitive results or better results with DIFFPOOL method. The only difference between them is that DIFFPOOL model averages the learned clusters' representations and StableGNN-NoCVD concatenates them by a consistent order, and then the aggregated embeddings are fed into a classifier. As the traditional MLP classifier needs the features of all samples should in a consistent order, the superior performance of StableGNN-NoCVD well validates that the learned representations $\mathbf{H}$ are in a consistent order across graphs. Moreover, we find that StableGNN consistently outperforms StableGNN-NoCVD. As the only difference between the two models is the CVD term, we can safely attribute the improvements to the distinguishing of causal variables by our proposed regularizer. Note that on some datasets, we could find that StableGNN-NoDVD cannot achieve satisfying results, however, when combined with the regularizer, it makes clear improvements. This phenomenon further validates the necessity of each component that we should conduct

5. Note that, for simplicity, in the following studies we mainly conduct analysis on StableSAGE, and StableGCN will get similar results. StableGNN will refer to StableeSAGE unless mentioned specifically.

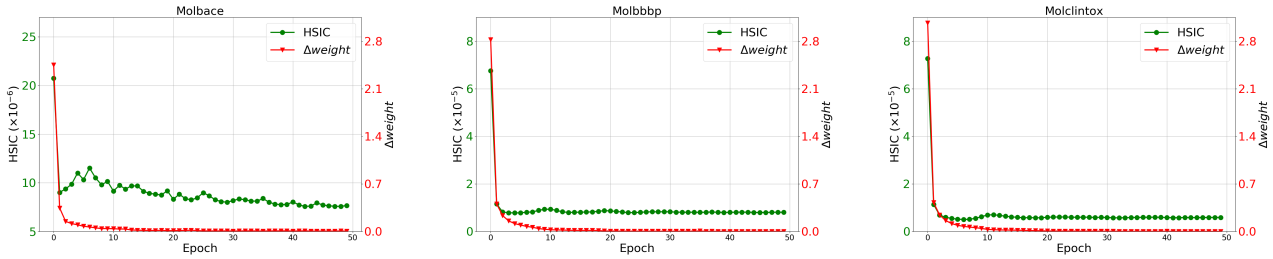


(a) Binary Classification (↑) (b) Multilabel Classification (↑) (c) Regression (↓)

Fig. 6: Ablation study on three real-world tasks.

TABLE 5: Performance of different decorrelation methods.

Method	Molhiv (↑)	Molbace (↑)	Molbbbp (↑)	MUTAG (↑)	Molclintox (↑)	Moltox21 (↑)	Molesol (↓)	Mollipo (↓)
GraphSAGE	75.78±2.19	78.51±1.72	66.16±0.97	79.78±0.7	88.60±2.44	68.88±0.59	1.8098±0.1220	0.7911±0.0147
GraphSAGE-Decorr	76.52±0.69	77.34±5.63	66.28±0.89	80.76±1.32	86.51±0.82	68.77±0.66	1.7889±0.1234	0.8024±0.0165
StableGNN-NoCVD	76.47±1.01	79.65±0.86	66.73±1.87	80.13±1.80	87.56±1.91	68.63±0.63	1.0819±0.0219	0.6971±0.017
StableGNN-NoCVD-Decorr	75.32±0.52	78.71±2.34	67.02±1.55	79.17±1.07	88.89±2.58	69.05±0.66	1.0377±0.0389	0.7171±0.0378
StableGNN-NoCVD-Distent	75.76±0.64	79.04±1.24	64.21±1.34	80.92+0.50	90.44±0.79	68.87±1.28	1.0729±0.0256	0.7171±0.0378
StableGNN	**77.63±0.79**	**80.73±3.98**	**68.47±2.47**	**82.13±0.32**	**90.96±1.93**	**69.14±0.24**	**1.022±0.0039**	**0.6971±0.0297**



(a) Molbace (b) Molbbbp (c) Molclintox

Fig. 7: Convergence rate analysis of CVD regularizer.

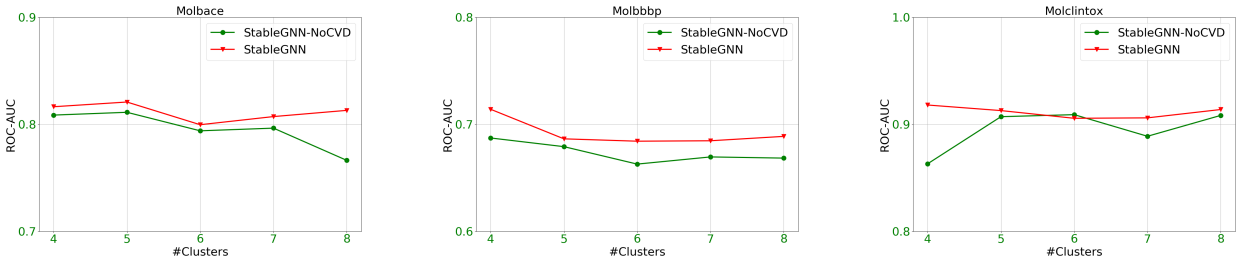
representation learning and causal inference jointly.

Comparison with Different Decorrelation Methods. As there may be other ways to decorrelate the variables in GNNs by sample reweighting methods, one question arises naturally: is our proposed decorrelation framework a more suitable strategy for graphs? To answer this question, we compare the following alternatives:

- GraphSAGE-Decorr: This method directly decorrelates each dimension of graph-level representations learned by GraphSAGE.
- StableGNN-NoCVD-Decorr: This method decorrelates each dimension of concatenated high-level representation $\mathbf{H}$ learned by StableGNN-NoCVD.
- StableGNN-NoCVD-Distent: This method forces the high-level representation learned by StableGNN-NoCVD to be disentangled by adding a HSIC regularizer to the overall loss.

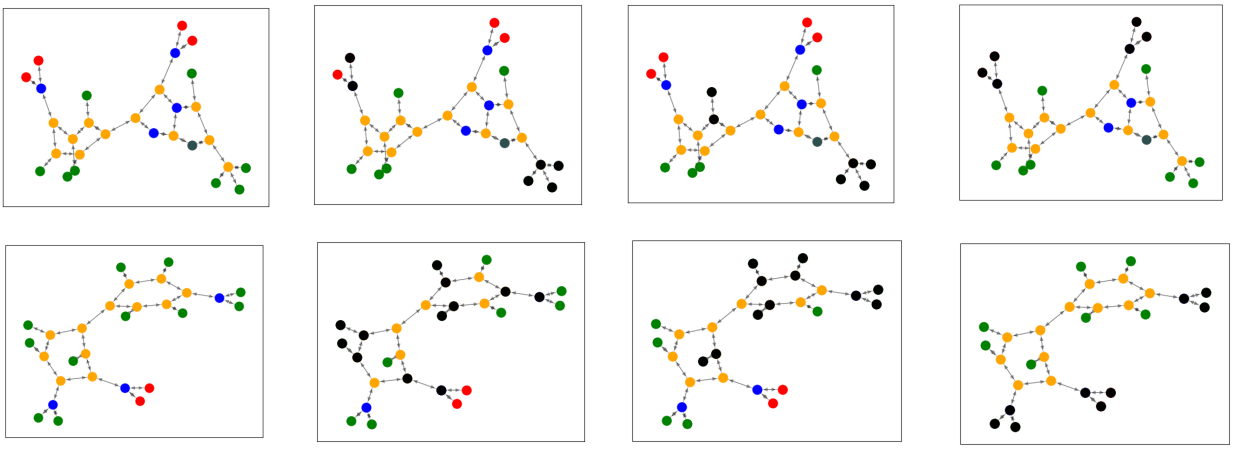
The results are shown in Table 5. Compared with these potential decorrelation methods, StableGNN achieves better results consistently, demonstrating that decorrelating the representations by the cluster-level granularity is a more suitable strategy for graph data. Moreover, we find that GraphSAGE-Decorr/StableGNN-NoCVD-Decorr shows worse performance than GraphSAGE/StableGNN-NoCVD, and the reason is that if we aggressively decorrelate single dimension of embeddings, it will inevitably break the intrinsic semantic meaning of original data. Furthermore, StableGNN-NoCVD-Distent forces the high-level representations to be disentangled, which changes the semantic implication of features, while StableGNN learns sample weights to adjust the data structure while the semantics of features are not affected. Overall, StableGNN is a general and effective framework compared with all the potential ways.

Convergence Rate and Parameter Sensitivity Analysis. We first analyze the convergence rate of our proposed causal variable distinguishing regularizer. We report the summation of HSIC value of all high-level representation pairs and the difference of learned weights between two consecutive epochs during the weights learning procedure in one batch on three relatively smaller datasets in Figure 7. As we can see, the weights learned by CVD regularizer could achieve convergence very fast while reducing the HSIC value significantly. In addition, we study the sensitivity of the number of high-level representations and report the performance of StableGNN-NoCVD and StableGNN based on the same pre-defined number of clusters in Figure 8. StableGNN outperforms StableGNN-NoCVD on almost all cases, which well demonstrates the robustness of our methods with the number of pre-defined clusters and the effectiveness of the proposed CVD regularizer. Note that our framework could learn the appropriate number of clusters in an end-to-end way, i.e., some clusters might not be used



(a) Molbace(↑) (b) Molbbbp(↑) (c) Molclintox(↑)

Fig. 8: Sensitivity of the pre-defined maximum number of clusters.



(a) Input graph (b) GCN (c) StableGCN-NoCVD (d) StableGCN

Fig. 9: Explanation instance for MUTAG dataset. The blue, green, red and yellow colors represent N, H, O, C atoms, respectively. The top important subgraphs selected by GNNExplainer are viewed in black color (keep the top 6 important edges). The picture is best viewed in color.

by the assignment matrix.

Interpretability Analysis. In Figure 9, we investigate explanations for graph classification tasks on MUTAG dataset. In the MUTAG example, colors indicate node features, which represent atoms. StableGNN correctly identifies chemical NO_2 and NH_2, which are known to be mutagenic [48] while baselines fail in. These cases demonstrate that our model could utilize more interpretable structures for prediction. Moreover, the difference of StableGCN-NoCVD between StableGCN indicates the necessity of integrating two components in our framework.

6 Conclusion and Future Work

In this paper, we study a practical but seldom studied problem: generalizing GNNs on out-of-distribution graph data. We analyze the problem in a causal view that the generalization of GNNs will be hindered by the spurious correlation among subgraphs. To improve the stability of existing methods, we propose a general causal representation learning framework, called StableGNN, which integrates graph high-level variable representation learning and causal effect estimation in a unified framework. Extensive experiments well demonstrate the effectiveness, flexibility, and interpretability of the StableGNN.

In addition, we believe that this paper just opens a direction for causal representation learning on graphs. As the most important contribution, we propose a general framework for causal graph representation learning: graph high-level variable representation learning and causal variable distinguishing, which can be flexibly adjusted for specific tasks. For example, besides molecules, we could adjust our framework to learn the causal substructures of proteins. The substructures of proteins play a crucial causal role in determining their properties and functions. Different substructures can confer distinct functionalities and characteristics on the proteins. To have a broader impact, we believe the idea could also spur causal representation learning in other areas, like object recognition [74], multi-modal data fusion [75], and automatic driving in wild environments [76].

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (No. U20B2045, 62192784,

62322203, 62172052, 62002029, 62141607, U1936219). This work was also supported in part by National Key R&D Program of China (No. 2018AAA0102004).

REFERENCES

[1] R. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "Gnnexplainer: Generating explanations for graph neural networks," in *NeurIPS*, 2019.

[2] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2008.

[3] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2016.

[4] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *ICLR*, 2017.

[5] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NeurIPS*, 2017, pp. 1024–1034.

[6] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," in *NeurIPS*, 2020.

[7] J. B. Lee, R. Rossi, and X. Kong, "Graph classification using structural attention," in *SIGKDD*, 2018, pp. 1666–1674.

[8] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *NeurIPS*, 2018.

[9] P. E. Pope, S. Kolouri, M. Rostami, C. E. Martin, and H. Hoffmann, "Explainability methods for graph convolutional neural networks," in *CVPR*, 2019, pp. 10 772–10 781.

[10] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *AAAI*, 2018.

[11] R. Liao, R. Urtasun, and R. Zemel, "A pac-bayesian approach to generalization bounds for graph neural networks," in *ICLR*, 2020.

[12] Y. Bengio, T. Deleu, N. Rahaman, R. Ke, S. Lachapelle, O. Bilaniuk, A. Goyal, and C. Pal, "A meta-transfer objective for learning to disentangle causal mechanisms," *arXiv preprint arXiv:1901.10912*, 2019.

[13] L. Engstrom, B. Tran, D. Tsipras, L. Schmidt, and A. Madry, "Exploring the landscape of spatial robustness," in *ICML*. PMLR, 2019, pp. 1802–1811.

[14] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *TEC*, vol. 23, no. 5, pp. 828–841, 2019.

[15] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," *arXiv preprint arXiv:1903.12261*, 2019.

[16] Y. Sun, X. Wang, Z. Liu, J. Miller, A. A. Efros, and M. Hardt, "Test-time training for out-of-distribution generalization," 2019.

[17] D. Krueger, E. Caballero, J.-H. Jacobsen, A. Zhang, J. Binas, D. Zhang, R. Le Priol, and A. Courville, "Out-of-distribution generalization via risk extrapolation (rex)," in *ICML*, 2021, pp. 5815–5826.

[18] X. Zhang, P. Cui, R. Xu, L. Zhou, Y. He, and Z. Shen, "Deep stable learning for out-of-distribution generalization," in *CVPR*, 2021, pp. 5372–5382.

[19] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman, "Building machines that learn and think like people," *Behavioral and brain sciences*, vol. 40, 2017.

[20] D. Lopez-Paz, R. Nishihara, S. Chintala, B. Scholkopf, and L. Bottou, "Discovering causal signals in images," in *CVPR*, 2017, pp. 6979–6987.

[21] W. Jin, R. Barzilay, and T. Jaakkola, "Hierarchical generation of molecular graphs using structural motifs," in *ICML*. PMLR, 2020, pp. 4839–4848.

[22] D. Bo, X. Wang, C. Shi, and H. Shen, "Beyond low-frequency information in graph convolutional networks," in *AAAI*, 2021.

[23] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European semantic web conference*. Springer, 2018, pp. 593–607.

[24] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," in *NeurIPS*, vol. 31, 2018, pp. 5165–5175.

[25] S. Fan, J. Zhu, X. Han, C. Shi, L. Hu, B. Ma, and Y. Li, "Metapath-guided heterogeneous graph neural network for intent recommendation," in *SIGKDD*, 2019, pp. 2478–2486.

[26] E. Pan and Z. Kang, "Beyond homophily: Reconstructing structure for graph-agnostic clustering," in *ICML*, 2023.

[27] S. Fan, X. Wang, C. Shi, E. Lu, K. Lin, and B. Wang, "One2multi graph autoencoder for multi-view graph clustering," in *WWW*, 2020, pp. 3070–3076.

[28] J. Lee, I. Lee, and J. Kang, "Self-attention graph pooling," in *ICML*. PMLR, 2019, pp. 3734–3743.

[29] H. Gao and S. Ji, "Graph u-nets," in *ICML*. PMLR, 2019, pp. 2083–2092.

[30] H. Li, X. Wang, Z. Zhang, and W. Zhu, "Ood-gnn: Out-of-distribution generalized graph neural network," *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[31] Y.-X. Wu, X. Wang, A. Zhang, X. He, and T.-S. Chua, "Discovering invariant rationales for graph neural networks," in *ICLR*, 2022.

[32] Y. Chen, Y. Zhang, Y. Bian, H. Yang, M. Kaili, B. Xie, T. Liu, B. Han, and J. Cheng, "Learning causally invariant representations for out-of-distribution generalization on graphs," in *NeurIPS*, vol. 35, 2022, pp. 22 131–22 148.

[33] H. Li, Z. Zhang, X. Wang, and W. Zhu, "Learning invariant graph representations for out-of-distribution generalization," in *Advances in Neural Information Processing Systems*, 2022.

[34] S. Fan, X. Wang, Y. Mo, C. Shi, and J. Tang, "Debiasing graph neural networks via learning disentangled causal substructure," *NeurIPS*, vol. 35, pp. 24 934–24 946, 2022.

[35] N. Yang, K. Zeng, Q. Wu, X. Jia, and J. Yan, "Learning substructure invariance for out-of-distribution molecular representations," in *NeurIPS*, 2022.

[36] B. Schölkopf, F. Locatello, S. Bauer, N. R. Ke, N. Kalchbrenner, A. Goyal, and Y. Bengio, "Toward causal representation learning," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 612–634, 2021.

[37] M. Arjovsky, L. Bottou, I. Gulrajani, and D. Lopez-Paz, "Invariant risk minimization," *arXiv preprint arXiv:1907.02893*, 2019.

[38] E. Rosenfeld, P. Ravikumar, and A. Risteski, "The risks of invariant risk minimization," in *ICLR*, 2020.

[39] P. Kamath, A. Tangella, D. Sutherland, and N. Srebro, "Does invariant risk minimization capture invariance?" in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 4069–4077.

[40] F. Qiao, L. Zhao, and X. Peng, "Learning to learn single domain generalization," in *CVPR*, 2020, pp. 12 556–12 565.

[41] T. Matsuura and T. Harada, "Domain generalization using a mixture of multiple latent domains," in *AAAI*, vol. 34, no. 07, 2020, pp. 11 749–11 756.

[42] H. Wang, Z. He, Z. C. Lipton, and E. P. Xing, "Learning robust representations by projecting superficial statistics out," *arXiv preprint arXiv:1903.06256*, 2019.

[43] K. Kuang, P. Cui, S. Athey, R. Xiong, and B. Li, "Stable prediction across unknown environments," in *SIGKDD*, 2018, pp. 1617–1626.

[44] K. Kuang, R. Xiong, P. Cui, S. Athey, and B. Li, "Stable prediction with model misspecification and agnostic distribution shift," in *AAAI*, 2020.

[45] J. Angrist and G. Imbens, "Identification and estimation of local average treatment effects," 1995.

[46] Z. Shen, P. Cui, K. Kuang, B. Li, and P. Chen, "Causally regularized learning with agnostic data selection bias," in *ACM MM*, 2018, pp. 411–419.

[47] Z. Shen, P. Cui, T. Zhang, and K. Kuang, "Stable learning via sample reweighting." in *AAAI*, 2020, pp. 5692–5699.

[48] D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, and X. Zhang, "Parameterized explainer for graph neural network," in *NeurIPS*, 2020.

[49] N. Keriven and G. Peyré, "Universal invariant and equivariant graph neural networks," *NeurIPS*, vol. 32, pp. 7092–7101, 2019.

[50] A. Leman and B. Weisfeiler, "A reduction of a graph to a canonical form and an algebra arising during this reduction," *Nauchno-Technicheskaya Informatsiya*, vol. 2, no. 9, pp. 12–16, 1968.

[51] J. Hainmueller, "Entropy balancing for causal effects: A multivariate reweighting method to produce balanced samples in observational studies," *Political Analysis*, vol. 20, no. 1, pp. 25–46, 2012.

[52] J. R. Zubizarreta, "Stable weights that balance covariates for estimation with incomplete outcome data," *Journal of the American Statistical Association*, vol. 110, no. 511, pp. 910–922, 2015.

[53] S. Athey, G. W. Imbens, and S. Wager, "Approximate residual balancing: De-biased inference of average treatment effects in high dimensions," *arXiv preprint arXiv:1604.07125*, 2016.

[54] L. Song, A. Smola, A. Gretton, K. M. Borgwardt, and J. Bedo, "Supervised feature selection via dependence estimation," in *ICML*, 2007, pp. 823–830.

[55] A. Gretton, O. Bousquet, A. Smola, and B. Schölkopf, "Measuring statistical dependence with hilbert-schmidt norms," in *International conference on algorithmic learning theory*. Springer, 2005, pp. 63–77.

[56] H. Zou, P. Cui, B. Li, Z. Shen, J. Ma, H. Yang, and Y. He, "Counterfactual prediction for bundle treatment," *NeurIPS*, vol. 33, 2020.

[57] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[58] L. Song, A. Smola, A. Gretton, J. Bedo, and K. Borgwardt, "Feature selection via dependence maximization." *Journal of Machine Learning Research*, vol. 13, no. 5, 2012.

[59] W. Lin, H. Lan, and B. Li, "Generative causal explanations for graph neural networks," in *ICML*, 2021.

[60] S. Ioffe and C. S. B. Normalization, "Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*.

[61] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[62] J. Huang and C. X. Ling, "Using auc and accuracy in evaluating learning algorithms," *TKDE*, vol. 17, no. 3, pp. 299–310, 2005.

[63] G. Landrum *et al.*, "Rdkit: Open-source cheminformatics," 2006.

[64] Z. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing, and V. Pande, "Moleculenet: a benchmark for molecular machine learning," *Chemical science*, vol. 9, no. 2, pp. 513–530, 2018.

[65] S. Ishida, T. Miyazaki, Y. Sugaya, and S. Omachi, "Graph neural networks with multiple feature extraction paths for chemical property estimation," *Molecules*, vol. 26, no. 11, p. 3125, 2021.

[66] A. K. Debnath, R. L. Lopez de Compadre, G. Debnath, A. J. Shusterman, and C. Hansch, "Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity," *Journal of medicinal chemistry*, vol. 34, no. 2, pp. 786–797, 1991.

[67] G. Landrum, "Rdkit: A software suite for cheminformatics, computational chemistry, and predictive modeling," 2013.

[68] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels." *Journal of Machine Learning Research*, vol. 12, no. 9, 2011.

[69] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt, "Efficient graphlet kernels for large graph comparison," in *Artificial intelligence and statistics*. PMLR, 2009, pp. 488–495.

[70] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *ICLR*, 2019. [Online]. Available: https://openreview.net/forum?id=ryGs6iA5Km

[71] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," in *CVPR*, 2017, pp. 5115–5124.

[72] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *ICML*. PMLR, 2019, pp. 6861–6871.

[73] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *ICML*. PMLR, 2018, pp. 5453–5462.

[74] M. Riesenhuber and T. Poggio, "Models of object recognition," *Nature neuroscience*, vol. 3, no. 11, pp. 1199–1204, 2000.

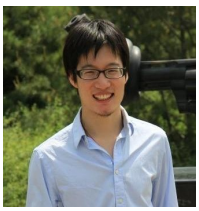
[75] Y. Wang, "Survey on deep multi-modal data analytics: Collaboration, rivalry, and fusion," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 17, no. 1s, pp. 1–25, 2021.

[76] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

Shaohua Fan received the Ph.D. degree in 2022 from Beijing University of Posts and Telecommunications. He was a visiting student at Mila for one year. Currently, he is a Postdoc researcher in the Department of Computer Science of Tsinghua University. His main research interests including graph mining, causal machine learning, and causal discovery. He has published several papers in major international conferences and journals, including NeurIPS, KDD, WWW and TNNLS etc.

Xiao Wang is an Associate Professor at Beihang University. He was an Associate Professor at Beijing University of Posts and Telecommunications. He received his Ph.D. degree from the School of Computer Science and Technology, Tianjin University, Tianjin, China, in 2016. He was a postdoctoral researcher in Tsinghua University. His current research interests include data mining, social network analysis, and machine learning. Until now, he has published more than 70 papers in conferences such as NeurIPS, AAAI, IJCAI, WWW, KDD, etc. and journals such as IEEE TKDE, IEEE Trans. on Cybernetics, etc.

Chuan Shi received the B.S. degree from the Jilin University in 2001, the M.S. degree from the Wuhan University in 2004, and Ph.D. degree from the ICT of Chinese Academic of Sciences in 2007. He joined the Beijing University of Posts and Telecommunications as a lecturer in 2007, and is a professor and deputy director of Beijing Key Lab of Intelligent Telecommunications Software and Multimedia at present. His research interests are in data mining, machine learning, and evolutionary computing. He has published more than 100 papers in refereed journals and conferences, such as TKDE, KDD, WWW, NeurIPS, and ICLR.



Peng Cui is an Associate Professor with tenure in Tsinghua University. He got his PhD degree from Tsinghua University in 2010. His research interests include causally-regularized machine learning, network representation learning, and social dynamics modeling. He has published more than 100 papers in prestigious conferences and journals in data mining and multimedia. He received ACM China Rising Star Award in 2015, and CCF-IEEE CS Young Scientist Award in 2018. He is now a Distinguished Member of ACM and CCF, and a Senior Member of IEEE.

Bai Wang received the B.S. degree from the Xian Jiaotong University, Xian, China and Ph.D. degree from the Beijing University of Posts and Telecommunications, Beijing, China. And she is currently a professor of computer science in BUPT. She was the director of Beijing Key Lab of Intelligent Telecommunications Software and Multimedia.