OOD-GNN: Out-of-Distribution Generalized Graph Neural Network

Haoyang Li, Xin Wang, *Member, IEEE,* Ziwei Zhang, *Member, IEEE,* Wenwu Zhu, *Fellow, IEEE*

Abstract—Graph neural networks (GNNs) have achieved impressive performance when testing and training graph data come from identical distribution. However, existing GNNs lack out-of-distribution generalization abilities so that their performance substantially degrades when there exist distribution shifts between testing and training graph data. To solve this problem, in this work, we propose an out-of-distribution generalized graph neural network (**OOD-GNN**) for achieving satisfactory performance on unseen testing graphs that have different distributions with training graphs. Our proposed **OOD-GNN** employs a novel nonlinear graph representation decorrelation method utilizing random Fourier features, which encourages the model to eliminate the statistical dependence between relevant and irrelevant graph representations through iteratively optimizing the sample graph weights and graph encoder. We further present a global weight estimator to learn weights for training graphs such that variables in graph representations are forced to be independent. The learned weights help the graph encoder to get rid of spurious correlations and, in turn, concentrate more on the true connection between learned discriminative graph representations and their ground-truth labels. We conduct extensive experiments to validate the out-of-distribution generalization abilities on two synthetic and 12 real-world datasets with distribution shifts. The results demonstrate that our proposed **OOD-GNN** significantly outperforms state-of-the-art baselines.

Index Terms—Graph Representation Learning, Graph Neural Networks, Out-of-Distribution Generalization.

1 Introduction

Graph structured data is ubiquitous in the real world, e.g., biology networks [1], social networks [2], molecular graphs [3], knowledge graphs [4], etc. Recently, deep learning models on graphs, especially graph neural networks (GNNs) [5–7], have increasingly emerged as prominent approaches for representation learning of graphs [8]. Significant methodological advances have been made in the field of GNNs, which have achieved promising performance in a wide variety of applications [9–12].

Despite their enormous success, the existing GNN approaches for graph representation learning generally assume that the testing and training graph data are independently sampled from the identical distribution, i.e., the I.I.D. assumption. In many real-world scenarios, however, it is difficult to guarantee this assumption to be valid. In particular, the testing distribution may suffer unobserved or uncontrolled shifts compared with the training distribution. For example, in the field of drug discovery, the prediction of biochemical properties of molecules is commonly trained on limited available experimental data, but the model needs to be tested on an extraordinarily diverse and combinatorially large universe of candidate molecules [13, 14]. The model performance of existing methods can be substantially degraded under distribution shifts due to the lack of out-of-distribution (**OOD**) generalization ability in realistic data splits [3, 15]. Therefore, it is of paramount importance to learn GNNs capable of out-of-distribution generalization and achieve relatively stable performances under distribution shifts, especially for some high-stake applications, e.g., medical diagnosis [16], criminal justice [17], financial analysis [18], and molecular prediction [3], etc.

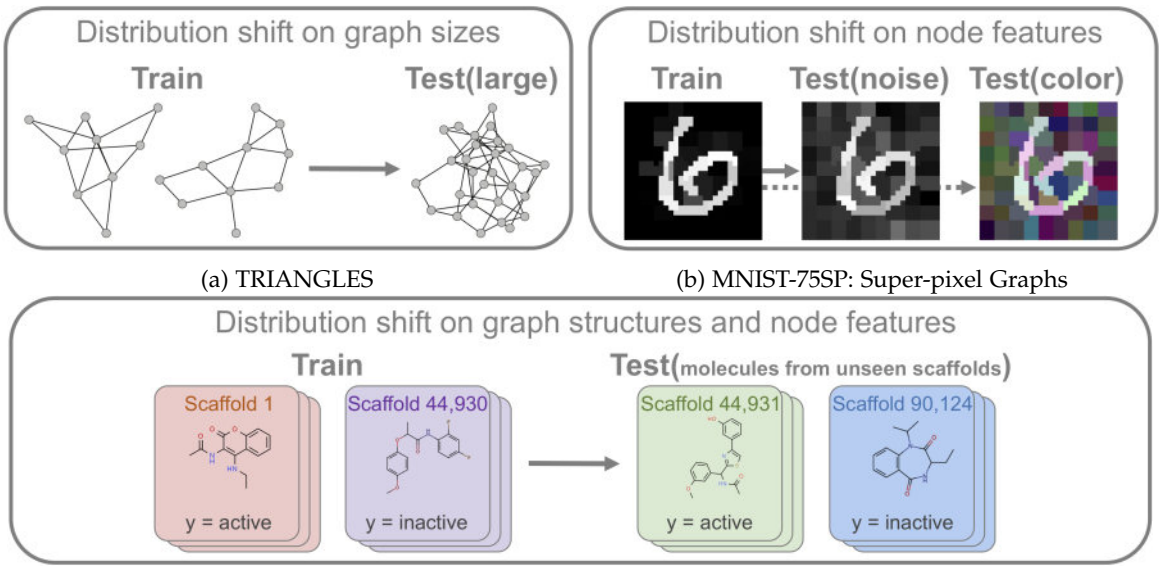
Some pioneering works [19–21] focus on the size generalization problem by testing on larger graphs than the training graphs. Besides size generalization, the capability of out-of-distribution generalization for GNNs is not explored until recently [22]. In out-of-distribution scenarios, when there exist complex heterogeneous distribution shifts, the performance of current GNN models can degrade substantially, which is mainly induced by the spurious correlations. The spurious correlations intrinsically come from the subtle correlations between irrelevant representations and relevant representations [23, 24]. For example, in the field of drug discovery (see Figure 1c), the GNN models trained on molecules with one group of scaffolds (two-dimensional structural frameworks of molecules) may learn the spurious correlations between the scaffolds and labels (i.e., whether some drug can inhibit HIV replication) [3, 15]. When tested on molecules with different scaffolds (out-of-distribution testing molecules), the existing GNN models may make incorrect predictions based on the spurious correlations.

In this paper, we propose to learn decorrelated graph representations through sample reweighting [25, 26] to eliminate the dependence between irrelevant and relevant representations, which is one of the major causes of degrading model performance under distribution shifts. However, learning decorrelated graph representations to improve out-of-distribution generalization for GNNs is fundamentally different from traditional methods and thus remains largely unexplored and challenging. Specifically, it poses the following challenges.

- GNNs fuse heterogeneous information from node features and graph structures such that the complex and

- *H. Li, X. Wang, Z. Zhang and W. Zhu are with the Department of Computer Science and Technology in Tsinghua University, Beijing, China. E-mail: lihy18@mails.tsinghua.edu.cn, {xin_wang, zwzhang, wwzhu}@tsinghua.edu.cn Corresponding authors: X. Wang and W. Zhu*

Manuscript received April 19, 2005; revised August 26, 2015.



(a) TRIANGLES (b) MNIST-75SP: Super-pixel Graphs

(c) OGB Molecule Dataset [27]. For validating OOD generalization, this dataset is split based on the scaffolds (i.e., two-dimensional structural frameworks) of molecules. The testing set consists of structurally distinct molecules with scaffolds that are not in the training set. Please refer to Section 4.1.2 for more details.

Fig. 1: Examples of out-of-distribution testing graphs under complex distribution shifts. Figure 1a denotes the models are trained on small graphs but tested on larger graphs. Figure 1b denotes the models trained with clean node features but tested with noisy features. Figure 1c represents a more realistic and challenging case, i.e., distribution shifts exist on both graph structures and node features.

unobserved non-linear dependencies among representations are much more difficult to be measured and eliminated than the linear cases for decorrelation of non-graph data.
- Although sample reweighting is effective on small datasets, for real-world large-scale graphs, it is inefficient or even infeasible to consistently learn a global weight for each graph in the dataset due to the high computational complexity and excessive storage consumption.

To tackle these challenges, we propose a novel out-of-distribution generalized graph neural network (**OOD-GNN**) capable of handling graph distribution shifts in complex and heterogeneous situations. In particular, we first propose to eliminate the statistical dependence between relevant and irrelevant graph representations of the graph encoder by a novel nonlinear graph representation decorrelation method utilizing random Fourier features [28–30], which scales linearly with the sample size and can get rid of unexpected spurious correlations. Next, to reduce computational complexity, we present a scalable global-local weight estimator to learn the sample weight for each graph. The local weights for a mini-batch of graphs and global weights for the entire graphs are optimized jointly to effectively maintain the consistency of weights over the whole graph dataset [29, 31, 32]. Finally, the parameters of the graph encoder and sample weights for graph representation decorrelation are optimized iteratively to learn discriminant graph representations for predictions.

We conduct extensive experiments on both synthetic graph datasets and well-known real-world graph benchmarks. The experimental results demonstrate that the representations learned from **OOD-GNN** can achieve substantial performance gains on the graph prediction tasks, including graph classification and regression, under distribution shifts.

The contributions of this paper are summarized as follows:

- We propose a novel out-of-distribution generalized graph neural network (**OOD-GNN**) capable of learning out-of-distribution (OOD) generalized graph representation under complex distribution shifts.
- We propose a nonlinear graph representation decorrelation method based on random Fourier features and sample reweighting. The decorrelated graph representations can substantially improve the out-of-distribution generalization ability in various OOD graph prediction benchmarks.
- We present a scalable global-local weight estimator to learn graph weights for the whole dataset consistently and efficiently. Extensive empirical results show that **OOD-GNN** greatly outperforms baselines on various graph prediction benchmarks under distribution shifts.

We review related works in Section 2. In Section 3, we describe the problem formulation and the details of our proposed **OOD-GNN**. Section 4 presents the experimental results including quantitative comparisons on both synthetic and real-world datasets, ablation studies, complexity analysis, hyper-parameter sensitivity, etc. Finally, we conclude our work in Section 5.

2 Related Works

Graph Neural Network. GNNs [5–7] have been attracting considerable attention in recent years because of their notable success in representing graph-structure data. They generally utilize a message-passing paradigm, which combines node features and graph topology to update node embeddings. To obtain the representation of the entire graph, graph

pooling [7, 33, 34] is adopted to summarize node embeddings. Many GNNs and their variants [35–38] have been proposed, achieving state-of-the-art performance on various graph tasks, including node classification [5], link prediction [39], and graph classification [7, 40]. Despite their successes, the performance of GNNs drops substantially when there are distribution shifts between training and testing graphs [3, 15, 27, 41]. The existing works largely ignore the out-of-distribution generalization ability of GNNs, which is crucial to realistic applications deployed in the wild.

Size generalization of GNNs. The main goal of size generalization is to make GNNs work well on testing graphs whose size distribution is different from that of training graphs [19–21, 42–44]. In these works, GNNs are usually trained on relatively small graphs and then generalize to larger graphs (or vice versa) with the help of attention mechanisms [19], self-supervised learning [20], causal modeling [21], etc. However, most existing methods only test on graphs of different sizes and ignore more realistic and challenging settings where the distribution shifts emerge in the graph topologies and node features.

The expressiveness of GNNs. The Weisfeiler-Lehman graph isomorphism test is most commonly used to measure the expressiveness power of GNNs [7, 36]. Assuming appropriate optimization, a more expressive GNN can achieve smaller error on the training data [45]. Some works [46, 47] study the generalization capability of GNNs over the training distribution with deriving generalization bounds. These works are orthogonal to out-of-distribution generalization, including unseen graph topological structures and features studied in this paper. The findings in [22] show that encoding task-specific non-linearities in the GNN architecture or features can improve the out-of-distribution generalization. However, it is largely unknown in practice that how to enhance the generalization ability of GNNs when there are distribution shifts between training and testing graphs.

Representation decorrelation. The spurious correlation between the irrelevant (non-critical) representations and labels is recognized as one major cause of model degradation under distribution shifts [48–51]. Some pioneering works adopt regularizers to penalize high correlation explicitly [48, 52, 53]. However, these methods could introduce a substantial computational overhead, yield marginal improvements, or require extra supervision to control the strength of the penalty. There are also some works learning decorrelated representations with sample reweighting [26, 54–56], which is shown effective in improving the generalization ability theoretically (e.g., SRDO [54]) and empirically (e.g., DWR [55]). However, most of these methods are proposed under linear settings. In contrast, GNNs fuse heterogeneous information from node features and graph topological structures so that there exist complex and unobserved non-linear dependencies among representations. The linear sample reweighting methods can not be applied to eliminate non-linear dependencies for the decorrelation of graph data. The effectiveness of non-linear decorrelation methods (e.g, ReBias [57], StableNet [29]) is validated on images recently. However, non-linear decorrelation on graphs remains largely unexplored.

TABLE 1: Notations.

Notation	Description
$\mathbf{G}^{tr}, \mathbf{G}^{te}$	The training and testing graph dataset
N^{tr}, N^{te}	The number of graphs in $\mathbf{G}^{tr}$ and $\mathbf{G}^{te}$
$\mathcal{G}, \mathcal{Y}$	The graph space and label space
$\mathcal{Z}$	The representation space
$\mathbf{G}, \mathbf{Y}$	The random variable of graph and label
$\mathbf{Z}$	The random variable of representation
Φ	The graph encoder from $\mathcal{G}$ to $\mathcal{Z}$
$\mathcal{R}$	The classifier from $\mathcal{Z}$ to $\mathcal{Y}$
G_n	A graph instance
$\mathbf{W}$	The graph weight vector
w_n	The weight for the graph G_n

3 METHOD

3.1 Notations and Problem Formulation

Let $\mathbf{G}^{tr} = \{G_n\}_{n=1}^{N^{tr}}$ and $\mathbf{G}^{te} = \{G_n\}_{n=1}^{N^{te}}$ be the training and testing graph dataset, which are under distribution shifts, i.e., $\mathrm{P}(\mathbf{G}^{tr}) \neq \mathrm{P}(\mathbf{G}^{te})$. $\mathbf{G}^{te}$ is unobserved in the training stage. A graph encoder $\Phi : \mathcal{G} \to \mathcal{Z}$ is a mapping from the input graph space $\mathcal{G}$ to a d-dimensional representation space $\mathcal{Z}$. In this work, we consider Φ as GNNs. $\mathcal{R} : \mathcal{Z} \to \mathcal{Y}$ is a classifier, mapping the representation space $\mathcal{Z}$ to the label space $\mathcal{Y}$. $\mathbf{G}, \mathbf{Z}, \mathbf{Y}$ denote sets of random variables in $\mathcal{G}, \mathcal{Z}, \mathcal{Y}$, respectively. Denote graph representations for $\mathbf{G}^{tr}$ as $\mathbf{Z} \subset \mathbb{R}^{N^{tr} \times d}$. $\mathbf{Z}_{n*}$ denotes the representation of the n-th graph and $\mathbf{Z}_{*i}$ is the random variable corresponding to the i-th dimension of $\mathbf{Z}$. Graph weights are $\mathbf{W} = \{w_n\}_{n=1}^{N^{tr}}$, where w_n is the weight for the n-th graph G_n in $\mathbf{G}^{tr}$ and we constrain $\sum_{n=1}^{N^{tr}} w_n = N^{tr}$. By jointly optimizing the graph encoder Φ, classifier $\mathcal{R}$, and graph weights $\mathbf{W}$, we aim to eliminate the statistical dependence of all dimensions in representation $\mathbf{Z}$ such that the predictor $\mathcal{R} \circ \Phi : \mathcal{G} \to \mathcal{Y}$ can achieve satisfactory generalization performance when testing on out-of-distribution graphs $\mathrm{P}(\mathbf{G}^{te})$. The key notations are summarized in Table 1. The formal definition of graph-level OOD generalization is as follows:

Definition 3.1 (Graph OOD Generalization)**.** Given the training set $\mathbf{G}^{tr}$, the goal is to learn an optimal graph predictor $\mathcal{R} \circ \Phi : \mathcal{G} \to \mathcal{Y}$ that can achieve the best generalization on the test graph data $\mathbf{G}^{te}$ under unknown distribution shifts, i.e., $\mathrm{P}(\mathbf{G}^{tr}) \neq \mathrm{P}(\mathbf{G}^{te})$.

Note that we expect to generalize the learned graph predictor on the OOD testing graphs instead of overfitting the training graphs, which can largely benefit the deployment of the graph predictor in the wild. The framework of our proposed method is shown in Figure 2.

3.2 Statistical Independence with Graph Reweighting

In this subsection, we describe the nonlinear graph representation decorrelation method to eliminate the statistical dependence of graph representations based on graph reweighting. Please refer to Appendix A for the background knowledge on the relationship of out-of-distribution (OOD) generalization and statistical independence, including the reason why the statistical dependence of graph representations should be eliminated for achieving OOD generalization.

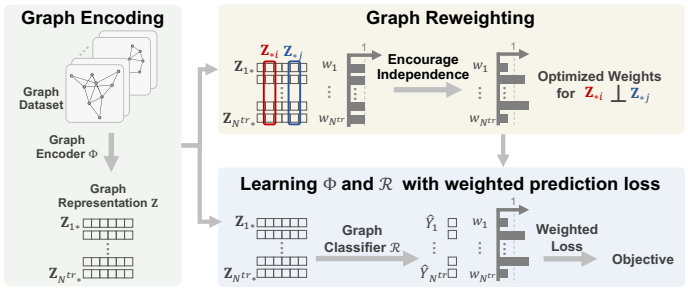


Fig. 2: The framework of the proposed method.

The correlation between relevant and irrelevant parts in representations is recognized as the main performance obstacle when $\mathrm{P}(\mathbf{G}^{tr}) \neq \mathrm{P}(\mathbf{G}^{te})$, i.e., OOD testing data [26, 58]. The relevant parts in representations denote the truly discriminant information to predict ground-truth labels, which are invariant under distribution shifts, e.g., the predictive functional groups of molecules. On the other hand, the irrelevant parts include non-informative features that could change across different domains, e.g., scaffold structure in predicting molecule functions. GNNs fuse available information from node features and graph topologies into a unified low-dimensional representation for each graph. So it is difficult or even infeasible to distinguish which dimensionality in the representation denotes relevant and irrelevant parts without extra supervision, which is unavailable and expensive to collect. Therefore, we propose to encourage the graph encoder to eliminate the statistical dependence of all dimensions in the graph representation. Note that we assume encouraging independence of all dimensions can benefit the OOD generalization and empirically observe this assumption is valid on the datasets and tasks in this work. Formally, we expect

$$\mathbf{Z}_{*i} \perp\!\!\!\perp \mathbf{Z}_{*j}, \forall i, j \in [1, d], i \neq j. \tag{1}$$

For measuring the independence between continuous random variables $\mathbf{Z}_{*i}$ and $\mathbf{Z}_{*j}$ in d-dimensional graph representation space $\mathcal{Z}$, it is inapplicable to resort to histogram-based measures unless d is small enough. So we introduce Hilbert-Schmidt Independence Criterion (HSIC) that can avoid the intractable explicit estimation of the joint distribution of the random variables and shows the strong empirical performance [59]. Specifically, consider a measurable, positive definite kernel $k_{\mathbf{Z}_{*i}}$ on the domain of random variable $\mathbf{Z}_{*i}$. Denote the corresponding Reproducing Kernel Hilbert Spaces (RKHS) by $\mathcal{H}_{\mathbf{Z}_{*i}}$. HSIC is defined as $\mathrm{HSIC}(\mathbf{Z}_{*i}, \mathbf{Z}_{*j}) := \|C_{\mathbf{Z}_{*i},\mathbf{Z}_{*j}}\|^2_{\mathrm{HS}}$, where $C_{\mathbf{Z}_{*i},\mathbf{Z}_{*j}}$ is the cross-covariance operator in the RKHS of $k_{\mathbf{Z}_{*i}}$ and $k_{\mathbf{Z}_{*j}}$. The independence can be determined as follows [60].

Proposition 1. *Assume* $\mathbb{E}[k_{\mathbf{Z}_{*i}}(\mathbf{Z}_{*i}, \mathbf{Z}_{*i})] < \infty$ *and* $\mathbb{E}[k_{\mathbf{Z}_{*j}}(\mathbf{Z}_{*j}, \mathbf{Z}_{*j})] < \infty$, *and* $k_{\mathbf{Z}_{*i}} k_{\mathbf{Z}_{*j}}$ *is a characteristic kernel, then*

$$\mathrm{HSIC}(\mathbf{Z}_{*i}, \mathbf{Z}_{*j}) = 0 \Leftrightarrow \mathbf{Z}_{*i} \perp\!\!\!\perp \mathbf{Z}_{*j}. \tag{2}$$

In practice, the finite-sample estimate of HSIC has been widely used [59]. However, it is infeasible to be utilized for training the graph encoder Φ on large-scale datasets (e.g., the OGBG-MOLHIV dataset in our experiments contains 41,127 graphs). The bottleneck lies in that the computational cost of HSIC grows as the batch size of training data increases. We therefore consider the squared Frobenius norm $\|\widehat{C}_{\mathbf{Z}_{*i},\mathbf{Z}_{*j}}\|^2_{\mathrm{F}}$, an analogue corresponding to the HSIC in Euclidean space[1] [29, 57], where $\widehat{C}_{\mathbf{Z}_{*i},\mathbf{Z}_{*j}}$ is the partial cross-covariance matrix defined as:

$$\widehat{C}_{\mathbf{Z}_{*i},\mathbf{Z}_{*j}} = \frac{1}{N^{tr}-1}\sum_{n=1}^{N^{tr}} \Big[\Big(f(\mathbf{Z}_{ni}) - \frac{1}{N^{tr}}\sum_{m=1}^{N^{tr}} f(\mathbf{Z}_{mi})\Big)^\top \cdot \Big(g(\mathbf{Z}_{nj}) - \frac{1}{N^{tr}}\sum_{m=1}^{N^{tr}} g(\mathbf{Z}_{mj})\Big)\Big], \tag{3}$$

where $\mathbf{Z}_{ni}$ and $\mathbf{Z}_{nj}$ denote the value of random variables $\mathbf{Z}_{*i}$ and $\mathbf{Z}_{*j}$ given the input graph G_n. $f(\cdot)$ and $g(\cdot)$ denote the random Fourier features concatenated from the Q selected functions from the random Fourier features function space:

$$\begin{aligned} f(\mathbf{Z}_{*i}) &:= (f_1(\mathbf{Z}_{*i}), f_2(\mathbf{Z}_{*i}), \ldots, f_Q(\mathbf{Z}_{*i})), \\ g(\mathbf{Z}_{*j}) &:= (g_1(\mathbf{Z}_{*j}), g_2(\mathbf{Z}_{*j}), \ldots, g_Q(\mathbf{Z}_{*j})), \end{aligned} \tag{4}$$

with $f_q(\mathbf{Z}_{*i}), g_q(\mathbf{Z}_{*j}) \in \mathcal{H}_{\mathrm{RFF}}, \forall q \in [1, Q]$. $\mathcal{H}_{\mathrm{RFF}} = \{h : x \to \sqrt{2}\cos(wx + \phi) | w \sim \mathcal{N}(0, 1), \phi \sim \mathrm{Uniform}(0, 2\pi)\}$ denotes the random Fourier features function space. The Eq. (4) means that we select Q functions from $\mathcal{H}_{\mathrm{RFF}}$ and concatenate their outputs for the calculation of the partial cross-covariance matrix $\widehat{C}_{\mathbf{Z}_{*i},\mathbf{Z}_{*j}}$ defined in Eq. (3). In a nutshell, we employ random Fourier feature (RFF) because it is an effective technique to approximate kernel-based independence test accurately and efficiently [29, 30, 61]. Note that as Q grows, the accuracy of independence judgement increases. And $Q = 5$ is solid enough to measure the independence of random variables in practice [29, 61].

Using the independence criterion above, we elaborate on graph reweighting which encourages the independence of the variables in graph representation. Define the graph weights $\mathbf{W} = \{w_n\}_{n=1}^{N^{tr}}$ where $w_n \in \mathbb{R}$ is the learnable weight for the n-th graph G_n in the training set. The graph weights can be directly utilized into Eq. (3), so the partial cross-covariance matrix can be calculated as:

$$\widehat{C}^{\mathbf{W}}_{\mathbf{Z}_{*i},\mathbf{Z}_{*j}} = \frac{1}{N^{tr}-1}\sum_{n=1}^{N^{tr}} \Big[\Big(w_n f(\mathbf{Z}_{ni}) - \frac{1}{N^{tr}}\sum_{m=1}^{N^{tr}} w_m f(\mathbf{Z}_{mi})\Big)^\top \cdot \Big(w_n g(\mathbf{Z}_{nj}) - \frac{1}{N^{tr}}\sum_{m=1}^{N^{tr}} w_m g(\mathbf{Z}_{mj})\Big)\Big]. \tag{5}$$

The learnable graph weight $\mathbf{W}$ participates in the optimization process to eliminate the dependence between representations to the greatest possible extent by minimizing the squared Frobenius norm of the partial cross-covariance matrix $\|\widehat{C}^{\mathbf{W}}_{\mathbf{Z}_{*i},\mathbf{Z}_{*j}}\|^2_{\mathrm{F}}$ in Eq. (5).

For the optimization, we iteratively optimize the graph weights $\mathbf{W}$, graph encoder Φ, and classifier $\mathcal{R}$:

$$\Phi^*, \mathcal{R}^* = \mathrm{argmin}_{\Phi,\mathcal{R}} \sum_{n=1}^{N^{tr}} w_n \ell\left(\mathcal{R} \circ \Phi\left(G_n\right), \mathbf{Y}_n\right), \tag{6}$$

$$\mathbf{W}^* = \mathrm{argmin}_{\mathbf{W}} \sum_{1 \le i < j \le d} \|\widehat{C}^{\mathbf{W}}_{\mathbf{Z}_{*i},\mathbf{Z}_{*j}}\|^2_{\mathrm{F}}, \tag{7}$$

where ℓ denotes the cross-entropy loss for graph classification tasks or mean squared error loss for graph regression tasks. The optimization of graph weights $\mathbf{W}$ in Eq. (7) encourages the graph encoder to generate the graph representations $\mathbf{Z} = \Phi(\mathbf{G})$, where each dimension keeps independent with

1. In a finite-dimensional Euclidean space, the Hilbert–Schmidt norm $\|\cdot\|_{\mathrm{HS}}$ is identical to the Frobenius norm.

others and thus eliminates the spurious correlations. The optimization of graph encoder Φ and classifier $\mathcal{R}$ in Eq. (6) based on the weighted graph datasets will lead to good performance on the specific prediction tasks.

3.3 Global-Local Graph Weight Estimator

Note that directly optimizing Eqs. (6)(7) requires all N^{tr} graph weights $\mathbf{W}$ and representations $\mathbf{Z}$ to calculate accurately $\widehat{C}^{\mathbf{W}}_{\mathbf{Z}_{*i},\mathbf{Z}_{*j}}$. Therefore, we need to load the entire dataset simultaneously for optimization, which is infeasible on large-scale datasets due to the high computational cost and excessive storage consumption. A straightforward alternative is to learn only graph representations and corresponding weights over a mini-batch of data. However, the consistency of the weights cannot be maintained since different mini-batches do not share information. Therefore, the dependence between different graph representation dimensions is hard to eliminate over the whole training dataset.

To tackle this problem, we present a novel scalable global-local weight estimator to achieve the balance of optimization efficiency and weight consistency, inspired by [29, 31, 32]. In essence, the motivation of adopting global weights is to save the global information over the whole dataset with constant size memory and keep the consistency of the learnable weights of all mini-batches. And we use the local weights to encourage the independence of different dimensions of the graph representations over a mini-batch. Next, we elaborate on the detailed designs of global and local weights, as well as their interactions.

Global weights. We maintain K groups of global representations $\mathbf{Z}^{(g)} = [\mathbf{Z}^{(g_1)}, \cdots, \mathbf{Z}^{(g_K)}]$ and the corresponding global weights $\mathbf{W}^{(g)} = [\mathbf{W}^{(g_1)}, \cdots, \mathbf{W}^{(g_K)}]$, where K is a hyper-parameter denoting the number of groups and the size of each group equals to the mini-batch, i.e., $\mathbf{Z}^{(g_k)} \in \mathbb{R}^{|\mathcal{B}|\times d}$ and $\mathbf{W}^{(g_k)} \in \mathbb{R}^{|\mathcal{B}|}$. $\mathbf{Z}^{(g_k)}$ is initialized to an all-zero matrix and $\mathbf{W}^{(g_k)}$ is uniformly initialized to an all-one vector $\mathbf{1}_{|\mathcal{B}|}$. They serve as the memory of the encoded graph representations and the corresponding weights from historical mini-batches during the training stage. Since these global representations and weights are shared across different mini-matches, they maintain a global summarization of the whole training dataset. The size of global weights only depends on the mini-batch size, which is a hyper-parameter and independent of the training dataset size.

Local weights. For each mini-batch $\mathcal{B}$ of the input graphs $\{G_n\}_{n=1}^{|\mathcal{B}|}$, we first calculate their graph representations $\mathbf{Z}^{(l)} = \{\mathbf{Z}^{(l)}_{n*}\}_{n=1}^{|\mathcal{B}|}$, $\mathbf{Z}^{(l)}_{n*} = \Phi(G_n)$ and uniformly initialize the local graph weights to an all-one vector $\mathbf{1}_{|\mathcal{B}|}$, i.e., $\mathbf{W}^{(l)} = (1, 1, \ldots, 1)$. Then, the local graph representations $\mathbf{Z}^{(l)}$ and weights $\mathbf{W}^{(l)}$ are concatenated with the K groups of global graph representations $\mathbf{Z}^{(g)}$ and weights $\mathbf{W}^{(g)}$ for optimization. We denote

$$\begin{aligned}\widehat{\mathbf{Z}} &= \left[\mathbf{Z}^{(g_1)}, \cdots, \mathbf{Z}^{(g_K)} \parallel \mathbf{Z}^{(l)}\right] \in \mathbb{R}^{(K+1)|\mathcal{B}|\times d},\\ \widehat{\mathbf{W}} &= \left[\mathbf{W}^{(g_1)}, \cdots, \mathbf{W}^{(g_K)} \parallel \mathbf{W}^{(l)}\right] \in \mathbb{R}^{(K+1)|\mathcal{B}|},\end{aligned} \tag{8}$$

where $[\cdot \parallel \cdot]$ is concatenation. Then, we calculate the weighted partial cross-covariance matrix in Eq. (5) using $\widehat{\mathbf{Z}}, \widehat{\mathbf{W}}$ and optimize the objective function. Using this estimator, the computational cost for each mini-batch is $O((K+1)|\mathcal{B}|)$, as opposed to $O(N^{tr})$ in directly optimizing Eqs. (6)(7).

Weights Update. At the end of each training iteration, we adopt a momentum update to dynamically update the global representations $\mathbf{Z}^{(g)}$ and weights $\mathbf{W}^{(g)}$ by the optimized local $\mathbf{Z}^{(l)}$ and $\mathbf{W}^{(l)}$:

$$\begin{aligned}\mathbf{Z}^{(g_k)} &\leftarrow \gamma_k \mathbf{Z}^{(g_k)} + (1-\gamma_k)\mathbf{Z}^{(l)},\\ \mathbf{W}^{(g_k)} &\leftarrow \gamma_k \mathbf{W}^{(g_k)} + (1-\gamma_k)\mathbf{W}^{(l)}.\end{aligned} \tag{9}$$

Eq. (9) shows the interaction of global and local representations and weights. Here $\gamma_k \in [0, 1)$ is a momentum coefficient for each group of global representations $\mathbf{Z}^{(g_k)}$ and weights $\mathbf{W}^{(g_k)}$. The global $\mathbf{Z}^{(g_k)}$ and $\mathbf{W}^{(g_k)}$ with a large γ_k serve as a long-term memory for global information over the whole training dataset, while those with a small γ_k serve as a short-term memory. Finally the global weights can be progressively updated and ensure the consistency of the whole graph dataset.

3.4 Training Procedure

The training procedure of our proposed **OOD-GNN** is shown in Algorithm 1.

Algorithm 1 The training procedure of **OOD-GNN**.

Input: A graph dataset $\mathbf{G} = \{G_n\}_{n=1}^{N}$
Output: Learned graph encoder Φ^* and classifier $\mathcal{R}^*$

1: **for** $e \leftarrow 1$ to Epoch **do**
2: **for** sampled minibatch $\mathcal{B} = \{G_n\}_{n=1}^{|\mathcal{B}|}$ **do**
3: Calculate $\mathbf{Z}^{(l)} = \{\mathbf{Z}^{(l)}_{n*}\}_{n=1}^{|\mathcal{B}|}$, $\mathbf{Z}^{(l)}_{n*} = \Phi(G_n)$
4: Initialize $\mathbf{W}^{(l)} = (1, 1, \ldots, 1)$
5: Concatenate global and local representations/weights as Eq. (8)
6: **for** $e' \leftarrow 1$ to Epoch_Reweight **do**
7: Optimize the graph weights by minimizing Eq. (7)
8: **end for**
9: Back propagate with weighted prediction loss as Eq. (6)
10: Update global representations and weights as Eq. (9)
11: **end for**
12: **end for**

At the training stage, we iteratively optimize the graph weights $\mathbf{W}$, graph encoder Φ, and classifier $\mathcal{R}$. Specifically, as shown in Algorithm 1, we first perform forward propagation for each sampled minibatch $\mathcal{B}$ to obtain the local graph representations $\mathbf{Z}^{(l)} = \{\mathbf{Z}^{(l)}_{n*}\}_{n=1}^{|\mathcal{B}|}$, $\mathbf{Z}^{(l)}_{n*} = \Phi(G_n)$ (line 3 in Algorithm 1) and uniformly initialize the local graph weights $\mathbf{W}^{(l)} = (1, 1, \ldots, 1)$ (line 4). To maintain consistency of the weights and improve efficiency, we concatenate the global representations and weights with local representations and weights to obtain $\widehat{\mathbf{Z}}$ and $\widehat{\mathbf{W}}$ (line 5). After that, we calculate the partial cross-covariance matrix $\widehat{C}^{\widehat{\mathbf{W}}}_{\widehat{\mathbf{Z}}_{*i},\widehat{\mathbf{Z}}_{*j}}$ and optimize the graph weights by minimizing the following objective function (line 7):

$$\mathbf{W}^{(l)*} = \operatorname{argmin}_{\mathbf{W}^{(l)}} \sum_{1\le i<j\le d} \|\widehat{C}^{\widehat{\mathbf{W}}}_{\widehat{\mathbf{Z}}_{*i},\widehat{\mathbf{Z}}_{*j}}\|_{\mathrm{F}}^2, \tag{10}$$

TABLE 2: The statistics of the datasets. #Graphs is the number of graphs in the dataset. Average #Nodes/#Edges are the average number of nodes and edges in a graph of the dataset, respectively. #Tasks denotes the dimensionality of output required for prediction. Task type includes binary classification, multi-classification, and regression. The various split methods for training/validation/testing dataset cover complex and realistic distribution shifts.

Category	Name	#Graphs	Average #Nodes	Average #Edges	#Tasks	Task Type	Split Method	Metric
Synthetic	TRIANGLES	4,000	15.6	48.9	1	Regression	Size	Accuracy
	MNIST-75SP	7,000	66.8	600.2	1	Multi-class.	Feature	Accuracy
Molecule and social datasets	COLLAB	5,000	74.5	2457.8	1	Multi-class.	Size	Accuracy
	PROTEINS	1,113	39.1	72.8	1	Binary class.	Size	Accuracy
	D&D	1,178	284.3	715.7	1	Binary class.	Size	Accuracy
Open Graph Benchmark OGBG-MOL*	TOX21	7,831	18.6	19.3	12	Binary class.	Scaffold	ROC-AUC
	BACE	1,513	34.1	36.9	1	Binary class.	Scaffold	ROC-AUC
	BBBP	2,039	24.1	26.0	1	Binary class.	Scaffold	ROC-AUC
	CLINTOX	1,477	26.2	27.9	2	Binary class.	Scaffold	ROC-AUC
	SIDER	1,427	33.6	35.4	27	Binary class.	Scaffold	ROC-AUC
	TOXCAST	8,576	18.8	19.3	12	Binary class.	Scaffold	ROC-AUC
	HIV	41,127	25.5	27.5	1	Binary class.	Scaffold	ROC-AUC
	ESOL	1,128	13.3	13.7	1	Regression	Scaffold	RMSE
	FREESOLV	642	8.7	8.4	1	Regression	Scaffold	RMSE

Next, we optimize the graph encoder Φ and classifier $\mathcal{R}$ by performing back propagation with weighted prediction loss (line 9):

$$\Phi^*, \mathcal{R}^* = \text{argmin}_{\Phi,\mathcal{R}} \sum_{n=1}^{|\mathcal{B}|} w_n \ell\left(\mathcal{R} \circ \Phi\left(G_n\right), \mathbf{Y}_n\right), \quad (11)$$

where $w_n = \mathbf{W}_n^{(l)*}$ is the optimized weight for the n-th graph in the minibatch $\mathcal{B}$. At the end of each iteration, the global representations and weights are updated by the optimized local graph representations and local graph weights (line 10).

At the testing stage, we directly adopt the optimized graph encoder Φ^* and classifier $\mathcal{R}^*$ to learn graph representations and conduct predictions.

4 Experiments

In this section, we empirically evaluate the effectiveness of the proposed **OOD-GNN** on both synthetic and real-world datasets and conduct ablation studies. More experimental results (including hyper-parameter sensitivity, training dynamic, weight distribution, time complexity, etc.) are also present and analyzed in detail.

4.1 Experimental Setup

4.1.1 Baselines

We compare our **OOD-GNN** with several representative state-of-the-art methods:

- GCN [5]: It is one of the most famous GNNs, following a recursive neighborhood aggregation (or message passing) scheme.
- GIN [7]: It is shown to be one of the most expressive GNNs in representation learning of graphs.
- GCN-virtual and GIN-virtual [15]: We also consider the variants of GCN and GIN augmented with virtual node, i.e., adding a node that is connected to all the nodes in the original graphs.
- FactorGCN [62]: It decomposes the input graph into several interpretable factor graphs for graph-level disentangled representations, which is a state-of-the-art disentangled GNN model for graph classification.
- Γ_{GIN} [21]: It adopts a causal model to learn size-invariant graph representations that can extrapolate to unseen test graphs with different graph sizes.
- PNA [63]: It takes multiple neighborhood aggregation schemes into account and generalizes several GNN models with different neighborhood aggregation schemes.
- TopKPool [64]: It propagates only part of the input and this part is not uniformly sampled from the input. It can thus select some local parts of the input graph and ignore the rest to summarize the graph representation.
- SAGPool [33]: It is a graph pooling method based on self-attention mechanism, which can be used to calculate attention scores and retain important nodes for graph-level representation.

4.1.2 Datasets

To cover more realistic and challenging cases of graph distribution shifts, we compare our method and baselines on both synthetic and real-world datasets:

- **Synthetic Datasets.**
  We use two synthetic datasets to evaluate the effectiveness of our proposed method, and examples of these datasets are shown in Figure 1a and 1b.
  (1) **TRIANGLES**. Counting the number of triangles in a graph is a common task that can be solved analytically but is challenging for GNNs. We first generate 4,000 random graphs using Erdős-Rényi model [65], i.e., ER(#Node, #Edge), where each graph has #Node nodes and the number of edges #Edge is randomly selected from [#Node, 2∗#Node]. We train on graphs containing 4 to 25 nodes, and test on graphs with 4 to 100 nodes. The node features are set as one-hot degrees. The dataset is split into 3,000/500/500 graphs used as training/validation/testing sets. The task is to predict the number of triangles in each graph. The number of classes is 10 (i.e., each graph has 1 to 10 triangles). Based

TABLE 3: Graph classification accuracy (%) on training and testing sets of two synthetic datasets. Test(large) denotes larger graph sizes in testing set and Test(noise)/Test(color) represent adding Gaussian noises/color noises respectively. In each column, the boldfaced score denotes the best result and the underlined score represents the second-best result. ± denotes standard deviation.

	TRIANGLES		**MNIST-75SP**		
	Train	Test(large)	Train	Test(noise)	Test(color)
GCN	28.3±0.6	21.3±1.9	51.7±1.0	26.5±1.4	27.0±1.3
GCN-virtual	32.4±0.6	17.0±1.8	55.1±2.3	26.0±1.5	26.1±1.8
GIN	34.7±0.7	22.2±1.9	67.6±0.8	27.9±2.5	34.3±4.4
GIN-virtual	34.2±0.6	17.6±1.7	66.7±0.9	25.7±2.9	33.4±1.2
FactorGCN	10.6±1.6	4.2±0.9	46.7±1.2	19.7±1.4	24.8±1.3
Γ_{GIN}	23.5±3.4	18.0±2.0	47.5±1.3	23.7±1.4	28.3±1.8
PNA	**43.7±3.6**	16.8±2.4	**83.0±0.9**	22.8±7.3	29.2±6.3
TopKPool	28.3±0.3	22.0±0.2	61.0±3.7	17.0±1.0	16.9±1.5
SAGPool	26.7±1.0	23.7±0.7	60.2±1.3	19.6±3.4	20.1±3.7
OOD-GNN	29.9±0.7	**25.1±0.8**	63.2±1.1	**31.5±0.9**	**38.5±1.5**

on this setting, there exist distribution shifts with regard to graph sizes between training and testing data.
(2) **MNIST-75SP**. Each graph in MNIST-75SP is converted from an image in MNIST [66] using super-pixels [67]. We randomly sample 7,000 images of MNIST and extract no more than 75 superpixels for each image to generate the graph. The node features are set as the super-pixel coordinates and intensity. There are 6,000 graphs used for training (of which the holdout 10% graphs for validation) and 1,000 graphs used for testing. The task is to classify each graph into the corresponding handwritten digit labeled from 0 to 9. To simulate distribution shifts with respect to graph features, we follow [19] and generate two testing graph datasets. For the first testing set, Test(noise), we add Gaussian noise, drawn from $\mathcal{N}(0, 0.4)$, to node features. For the second testing set, Test(color), we colorize images by adding two more channels and add independent Gaussian noise, drawn from $\mathcal{N}(0, 0.4)$, to each channel. The graph structures (adjacency matrices) are not changed for testing graphs.

- **Real-world Datasets.**
(1) **Molecule and social datasets**. We consider three commonly used graph classification benchmarks: **COLLAB** [68], **PROTEINS** [69], and **D&D** [70]. Following [19], these datasets are split based on the size of each graph. **D&D$_{200}$** and **D&D$_{300}$** denote the two datasets whose maximum graph size in the training set is 200 and 300, respectively. All the methods are trained on smaller graphs and tested on unseen larger graphs. Specifically, **COLLAB** is derived from 3 public collaboration datasets, i.e., High Energy Physics, Condensed Matter Physics, and Astro Physics. We train on graphs with 32 to 35 nodes and test on graphs with 32 to 492 nodes. **PROTEINS** is a protein dataset. We train on graphs with 4 to 25 nodes and test on graphs with 6 to 620 nodes. **D&D** is also a dataset that consists of proteins. We consider two types of splitting methods, termed **D&D$_{200}$** and **D&D$_{300}$**. For **D&D$_{200}$**, we train on graphs with 30 to 200 nodes and test on graphs with 201 to 5,748 nodes. For **D&D$_{300}$**, we train on 500 graphs with 30 to 300 nodes and test on other graphs with 30 to 5,748 nodes.
(2) **Open Graph Benchmark (OGB)** [15]. We consider 9 graph property prediction datasets from a benchmark of distribution shifts **OGBG-MOL**$*$ in Open Graph Benchmark (OGB), i.e., TOX21, BACE, BBBP, CLINTOX, SIDER, TOXCAST, HIV, ESOL, FREESOLV. The task is to predict the target molecular properties as accurately as possible. We adopt the default scaffold splitting procedure, namely splitting the graphs based on their two-dimensional structural frameworks. The scaffolds denote the basis structures of molecules, which are the frequently occurring common subgraphs in the datasets. Although such subgraphs do not provide the truly discriminant information to predict ground-truth labels, they may form the spurious correlations with labels. Therefore, this scaffold splitting strategy separate structurally different molecules into different subsets, which provides a more realistic and challenging scenario of out-of-distribution generalization. Figure 1c shows some examples of the dataset.

4.1.3 *Implementation Details*

The number of epochs (i.e., Epoch in Algorithm 1) is set to 100. The batch size is chosen from {64, 128, 256}. The learning rate is chosen from {0.0001, 0.001, 0.005}. The number of epochs of learning graph weights (i.e., Epoch_Reweight in Algorithm 1) is set to 20. The dimensionality of the representations and hidden layers d is chosen from {128, 300} for OGB, and {64, 256} for other datasets. While our method is general and compatible with most representative GNNs, we focus on using GIN [7] as the graph encoder in our experiments, since it is shown to be one of the most expressive GNNs in graph classification, and the number of layers is chosen from [2, 6]. We set $Q = 1$ to sample random Fourier features. The ℓ^2-norm is adopted on the weights to prevent degenerated solutions. The number of groups of global representations and weights $K = 1$ with the momentum coefficient $\gamma = 0.9$ in the updating step. The classifier $\mathcal{R} : \mathcal{Z} \to \mathcal{Y}$ is realized by a two-layer MLP. Note that for OGB datasets, we adopt the default validation set provided by the benchmark for fair comparisons, and use the holdout validation set from training set for the other datasets. Specifically, we hold out 10% training data as validation set for COLLAB, PROTEINS, and D&D. The holdout validation set follows the same distribution as the training set. We report the mean values with standard deviations of 10 repeated experiments.

TABLE 4: Graph classification accuracy (%) on the **testing set** of **OOD-GNN** and baselines. Our **OOD-GNN** outperforms the baselines significantly on all graph classification benchmarks, indicating its superiority against graph size distribution shifts. The best result and the second-best result for each dataset are in bold and underlined, respectively.

	$\textbf{COLLAB}_{35}$	$\textbf{PROTEINS}_{25}$	$\textbf{D\&D}_{200}$	$\textbf{D\&D}_{300}$
# Train/Test graphs	500/4500	500/613	462/716	500/678
#Nodes Train	32-35	4-25	30-200	30-300
#Nodes Test	32-492	6-620	201-5748	30-5748
GCN	65.9±3.4	75.1±2.2	29.2±8.2	71.9±3.6
GCN-virtual	61.5±1.6	70.4±3.7	41.6±8.0	71.6±4.4
GIN	55.5±4.9	74.0±2.7	43.0±8.3	67.8±4.3
GIN-virtual	54.8±2.7	66.0±7.5	46.7±4.5	72.1±4.3
FactorGCN	51.0±1.3	63.5±4.8	42.3±3.1	55.9±1.6
Γ_{GIN}	65.2±2.3	62.6±3.6	59.6±4.9	74.2±2.8
PNA	59.6±5.5	71.4±3.4	47.3±6.8	70.1±2.1
TopKPool	52.8±1.0	64.9±3.0	34.6±5.6	69.3±3.6
SAGPool	67.0±1.7	76.2±0.7	54.3±5.0	78.4±1.1
OOD-GNN	**67.2±1.8**	**78.4±0.9**	**60.3±4.5**	**80.1±1.0**

4.2 Results on Synthetic Graphs

The results on TRIANGLES and MNIST-75SP are reported in Table 3. On TRIANGLES, there exist distribution shifts on the graph sizes. **OOD-GNN** consistently achieves the best testing performance compared with other baselines on the out-of-distribution testing graphs, demonstrating the OOD generalization capability of our method. The accuracy of a strong baseline PNA on training graphs is impressive but drops significantly on the OOD testing graphs. FactorGCN, as a disentangled graph representation learning method, decomposes the input graph into several independent factor graphs so that it may change the semantic implication of representations into these implicit factors and affect the performance. In contrast, **OOD-GNN** learns graph weights so that the semantic of the graph representations will not be affected, leading to better generalization ability.

On MNIST-75SP, there exist distribution shifts on graph features, i.e., graphs in the testing datasets have larger noises. **OOD-GNN** achieves the best performance consistently compared with other methods. For this dataset, each graph consists of super-pixel nodes and edges that are formed based on the spatial distance between super-pixel centers. Therefore, the graph topological structures are relatively more discriminative than node features in making predictions. Traditional GNNs fuse heterogeneous information from both graph topological structures and features into unified graph representations, so these baselines may learn the spurious correlations, leading to poor generalization performance. Our method performs well on OOD testing graphs. One plausible reason is that when complex non-linear dependencies between graph structures and features are eliminated, our method is more likely to learn the true connections between relevant representations (i.e., informative graph topological structures) and labels, and conduct inference according to them only, thus generalize better.

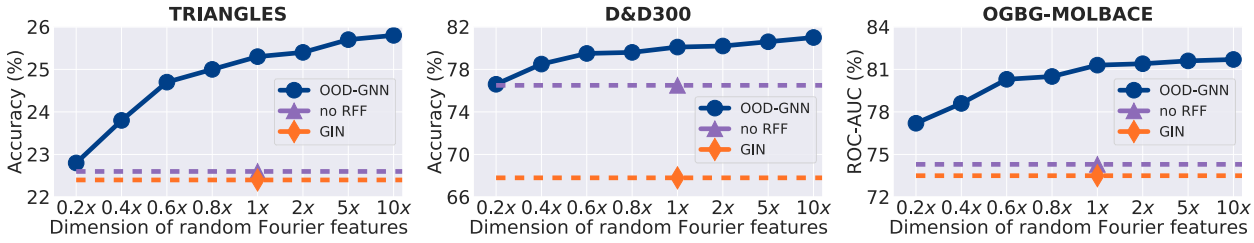
4.3 Results on Real-world Graphs

On real-world molecule and social datasets (i.e., COLLAB, PROTEINS, and D&D), the training and testing graphs are split by graph sizes, i.e., our method and baselines are trained on small graphs and tested on larger graphs. The results are presented in Table 4. **OOD-GNN** consistently yields the best testing performance on all the datasets. In particular, **OOD-GNN** improves over the strongest baselines by 2.2% and 1.7% in PROTEINS_{25} and $\text{D\&D}_{300}$ respectively. Our model achieves the best OOD generalization performance under size distribution shifts by encouraging independence between relevant and irrelevant representations. The results of baselines degrade due to the spurious correlations between irrelevant representations and labels. For example, each graph in the COLLAB dataset corresponds to an ego-network of different researchers from one field, and the label denotes the corresponding research field. The truly predictive representations are from the graph topological structures. If the GNN models tend to learn spurious correlations between graph sizes and labels but not focus more on the truly predictive graph structures, they will fail to make correct predictions on larger OOD testing graphs.

The graph classification results on nine Open Graph Benchmark (OGB) datasets are shown in Table 5. The datasets are split based on the scaffold, i.e., the two-dimensional structural framework. So the distribution shifts between training and testing graphs exist on the graph topological structure and features, leading to a more challenging scenario. None of the baselines is consistently competitive across all datasets, as opposed to our proposed method. Notice that adding virtual nodes to GCN or GIN is not a promising improvement for generalization since it can provide performance gains on some datasets but fail on the others. FactorGCN shows poor results, possibly because it enforces the decomposition of the input graphs into several independent factor graphs for disentanglement, which is hard to achieve without sufficient supervision. Γ_{GIN} is a size generalization method and also performs poorly on OGB datasets. PNA is proposed to address the size generalization problem but still fails under the more complex distribution shifts. TopKPool selects some local parts of the input graph and ignores the others. The strongest baseline on molecule and social datasets, i.e., SAGPool, pools the nodes with self-attention mechanism. However, the accurate selection for TopKPool and calculation of attention scores for SAGPool are easily affected by the spurious correlations on OOD test graphs and therefore also fail to generalize. In contrast, **OOD-GNN** shows a strong capability of out-of-distribution generalization when the input graphs have complicated

TABLE 5: Results on nine Open Graph Benchmark (OGB) datasets. We report the ROC-AUC (%) for classification tasks and RMSE for regression tasks with the standard deviation on the **test set** of all methods. None of the baseline methods is consistently competitive across all datasets, while our proposed method shows impressive performance. (↑) means that higher values indicate better results, and (↓) represents the opposite.

	TOX21	BACE	BBBP	CLINTOX	SIDER	TOXCAST	HIV	ESOL	FREESOLV
Metric	ROC-AUC (↑)							RMSE (↓)	
GCN	75.3±0.7	79.2±1.4	68.9±1.5	91.3±1.7	59.6±1.8	63.5±0.4	76.1±1.0	1.11±0.03	2.64±0.24
GCN-virtual	77.5±0.9	68.9±7.0	67.8±2.4	88.6±2.1	59.8±1.5	66.7±0.5	76.0±1.2	1.02±0.10	2.19±0.12
GIN	74.9±0.5	73.0±4.0	68.2±1.5	88.1±2.5	57.6±1.4	63.4±0.7	75.6±1.4	1.17±0.06	2.76±0.35
GIN-virtual	77.6±0.6	73.5±5.2	69.7±1.9	84.1±3.8	57.6±1.6	66.1±0.5	77.1±1.5	1.00±0.07	2.15±0.30
FactorGCN	57.8±2.1	70.0±0.6	54.1±1.1	64.2±2.1	53.3±1.7	51.2±0.8	57.1±1.5	3.39±0.15	5.69±0.32
Γ_{GIN}	52.3±1.1	54.5±0.9	51.8±1.5	52.2±1.1	51.3±1.1	50.7±0.5	51.3±0.9	4.15±0.10	7.34±0.12
PNA	71.5±0.5	77.4±2.1	66.2±1.2	81.2±2.0	59.6±1.1	60.6±0.2	79.1±1.3	0.94±0.02	2.92±0.16
TopKPool	75.6±0.9	76.9±2.4	68.6±1.1	86.9±1.1	60.6±1.5	64.7±0.1	76.7±1.1	1.17±0.03	2.08±0.10
SAGPool	74.7±3.1	76.6±1.0	69.3±2.1	88.7±1.0	61.3±1.3	64.8±0.2	77.7±1.3	1.22±0.05	2.28±0.12
OOD-GNN	**78.4±0.8**	**81.3±1.2**	**70.1±1.0**	**91.4±1.3**	**64.0±1.3**	**68.7±0.3**	**79.5±0.9**	**0.88±0.05**	**1.81±0.14**



(a) TRIANGLES (b) D&D$_{300}$ (c) OGBG-MOLBACE

Fig. 3: Ablation study results of our method. The blue curves with circle markers show that as dimensionality of random Fourier features increases, the generalization performance of **OOD-GNN** improves. The purple markers show that if we remove random Fourier features and only eliminate linear correlation, the performance drops significantly. The orange markers represent the results of GIN, the graph encoder baseline in our method.

structures, especially for the large-scale real-world graphs.

4.4 Ablation Studies

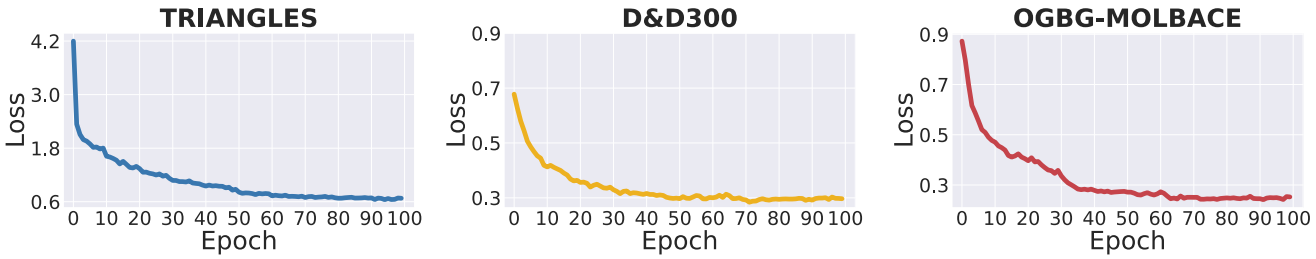
We perform ablation studies over a number of key components of our method to analyze their functionalities more deeply. Specifically, we compare **OOD-GNN** with the following two variants: (1) **Variant 1**: it sets the dimensionality of random Fourier features to different values. (2) **Variant 2**: it removes all the random Fourier features. For simplicity, we only report the results on one synthetic dataset (i.e., TRIANGLES) and two real-world datasets (i.e., D&D$_{300}$ and OGBG-MOLBACE), while the results on other datasets show similar patterns.

Variant 1 exploits the effect of different dimensions of random Fourier features. Note that our method adopts random Fourier features (see Eq. (4)), which sample from Gaussian to learn the graph weights and encourage the independence of representations. It is shown in [61] that if sampling more random Fourier features (i.e., when Q in Eq. (4) increases), the learned graph representations will be more independent. However, there exists a trade-off between independence and computational efficiency since the more random Fourier features are sampled, the higher the computational cost becomes. When the computational resources are extremely limited, it is also feasible to randomly select part of the dimensions in graph representations to calculate the dependence. In Figure 3, the x-axis represents the dimensionality of random Fourier features compared to graph representations, e.g., "2x" indicates $Q = 2$ in Eq. (4), while "0.2x" means we randomly select 20% dimensions of graph representations. We observe from Figure 3 that as the dimensionality of random Fourier features increases, the performance on OOD testing graphs grows consistently, which demonstrates that eliminating the statistical dependence between different dimensions of the graph representations will encourage the independence between relevant and irrelevant representations and lead to better out-of-distribution generalization ability.

Variant 2 removes all the random Fourier features and the optimization in Eqs. (6)(7) will degenerate to linear cases, i.e., only eliminating linear correlation rather than encouraging independence between different dimensions of graph representations. In Figure 3, this variant is termed as "no RFF". We can observe a clear performance drop for this variant, demonstrating that the complex non-linear dependencies are common in the graph representations. By eliminating non-linear dependence between representations, the GNNs will be encouraged to learn true connections between the input graphs and the corresponding labels.

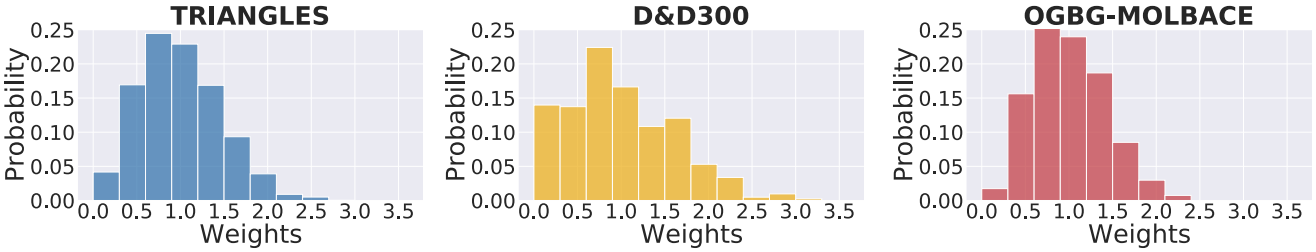
4.5 Training Dynamic

We can observe the convergence of our proposed method empirically, although Eqs. (6)(7) are iteratively optimized. In Figure 4 (a)(b)(c), we show the weighted prediction loss in the training process on TRIANGLES, D&D$_{300}$, and OGBG-MOLBACE, respectively. The loss converges in no more than 100 epochs to about 0.67, 0.30, and 0.25 on the three datasets, respectively. The results on the other datasets show similar patterns.



(a) Training loss on TRIANGLES. (b) Training loss on D&D$_{300}$. (c) Training loss on OGBG-MOLBACE.

Fig. 4: The weighted prediction loss in the training process on three datasets.



(a) Weights distribution on TRIANGLES. (b) Weights distribution on D&D$_{300}$. (c) Weights distribution on OGBG-MOLBACE.

Fig. 5: The distribution of the learned graph weights after training on three datasets.

4.6 Weights Distribution

In Figure 5 (a)(b)(c), to further investigate the effectiveness of the graph reweighting, we show the distribution of the learned graph weights on TRIANGLES, D&D$_{300}$, and OGBG-MOLBACE when the training is finished. The results show that our proposed method learns non-trivial weights, and the weights distribution is slightly different across different datasets.

4.7 Time Complexity

Our method is not only effective but efficient to learn out-of-distribution generalized graph representation under complex distribution shifts. The time complexity of our method is $O(|E|\,d+|V|\,d^2+K|\mathcal{B}|d^2)$, where $|V|$, $|E|$ denotes the total number of nodes and edges in the graphs, d is the dimensionality of the representation, K is the number of groups of global weights, and $|\mathcal{B}|$ is the batch size. Specifically, the time complexity of the graph encoder GIN is $O(|E|\,d+|V|\,d^2)$ and the optimization of graph weights in Eq. (7) has $O(K|\mathcal{B}|d^2)$ complexity. As a comparison, the time complexity of GIN, our backbone GNN, is $O(|E|\,d+|V|\,d^2)$, i.e., our time complexity is on par since d, K, and $|\mathcal{B}|$ are small constants that are unrelated to the dataset size.

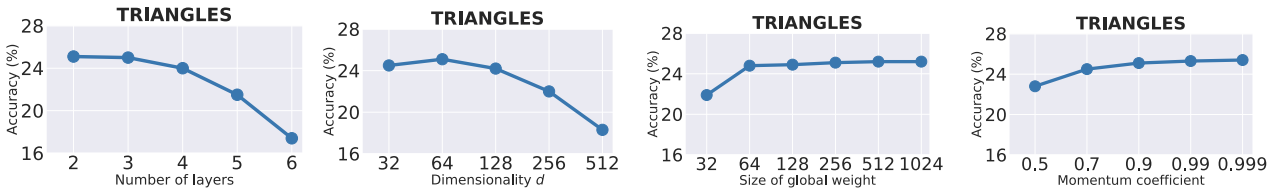
Note that the global-local weight estimator plays an important role in the efficiency. Without this module, the time complexity for optimizing consistent graph weights on the whole dataset can reach $O(N^{tr}d^2)$, which is related to the number of training graphs N^{tr} and intractable for large-scale real-world graph datasets. Thanks to the global-local weight estimator, the time complexity reduces to $O(K|\mathcal{B}|d^2)$ which is comparable with the backbone and will not induce higher computational costs. For the TRIANGLE dataset, our method with and without the global-local weight estimator, take 32s and 175s to optimize graph weights in the training stage, respectively, demonstrating the efficiency of our global-local weight estimator in practice.

4.8 Number of Parameters

The parameters of our method consist of two parts, i.e., the graph encoder and graph weights. The former is determined by the graph encoder GNN architecture, which is GIN in our setting. The latter is determined by the number of graphs. Taking the OGBG-MOLBACE dataset for example, the number of parameters of our method is about 0.9M if we set the number of message-passing layers as 5 and the dimensionality of the representations as 300. Notice that our method has comparable or fewer parameters than the baselines. For the OGBG-MOLBACE dataset with the same hyper-parameter settings, GIN and PNA (two baselines in the experiments) have 0.9M and 6.0M parameters, respectively. Nevertheless, our method achieves impressive out-of-distribution generalization performance against the baselines.

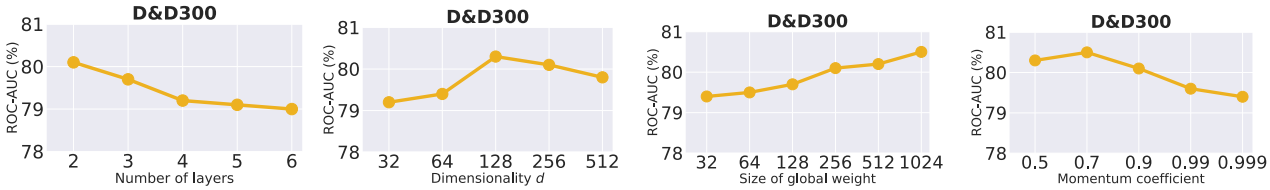
4.9 Hyper-parameter Sensitivity

We investigate the sensitivity of hyper-parameters of our method, including the number of message-passing layers in the graph encoder, the dimensionality of the representations d, the size of global weights, and the momentum coefficient γ in updating global weights. For simplicity, we only report the results on TRIANGLES (see Figure 6), D&D$_{300}$ (see Figure 7), and OGBG-MOLBACE (see Figure 8), while the results on other datasets show similar patterns. From Figures 6-8, we observe that the performance relies on an appropriate choice of the number of message-passing layers of the graph encoder. Since the task of counting triangles is relatively simple, the graph encoder with two message-passing layers



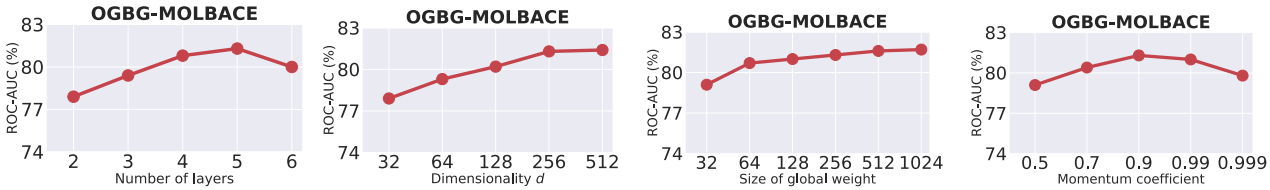
(a) Number of layers. (b) Dimensionality d. (c) Size of global weights. (d) Momentum coefficient.

Fig. 6: The analyses of different hyper-parameters on TRIANGLES dataset.



(a) Number of layers. (b) Dimensionality d. (c) Size of global global weights. (d) Momentum coefficient.

Fig. 7: The analyses of different hyper-parameters on D&D$_{300}$ dataset.



(a) Number of layers. (b) Dimensionality d. (c) Size of global weights. (d) Momentum coefficient.

Fig. 8: The analyses of different hyper-parameters on OGBG-MOLBACE dataset.

is good enough on TRIANGLES, while five layers are needed to achieve the best performance on OGBG-MOLBACE. When the number of layers of graph encoder is small, the model has limited capacity and may not be able to fuse enough information from neighbors. On the other hand, a very large number of layers could lead to the over-smoothing problem [71]. Besides, the optimal dimensionality of the representations d for TRIANGLES is relatively smaller than that for D&D$_{300}$ and OGBG-MOLBACE.

In addition, as the size of global weights increases, the performance is improved. The global representations and weights can help to learn consistent graph sample weights on the whole dataset and therefore improve the generalization ability of the model.

Finally, we find that the momentum coefficient γ also has a slight influence on the performance. A large γ will make the update of global representations and weights slower, and a small one will accelerate the update, corresponding to emphasizing long-term and short-term memory, respectively.

5 Conclusions

In this paper, we propose a novel out-of-distribution generalized graph neural network (**OOD-GNN**) to solve the problem of generalization of GNNs under complex and heterogeneous distribution shifts. We propose a nonlinear graph representation decorrelation method by utilizing random Fourier features and sample reweighting, so that the learned representations of **OOD-GNN** are encouraged to eliminate the statistical dependence between the representations. We further present a scalable global-local weight estimator, which can learn graph weights for the whole dataset consistently and efficiently. Extensive experiments on both synthetic and real-world datasets demonstrate the superiority of our method against state-of-the-art baselines for out-of-distribution generalization.

Acknowledgments

The authors would like to thank Xingxuan Zhang for valuable discussions. This work was supported in part by the National Key Research and Development Program of China No. 2020AAA0106300 and National Natural Science Foundation of China No. 62250008, No. 62102222.

References

[1] A.-L. Barabasi and Z. N. Oltvai, "Network biology: understanding the cell's functional organization," *Nature reviews genetics*, vol. 5, no. 2, pp. 101–113, 2004.

[2] D. Easley, J. Kleinberg *et al.*, *Networks, crowds, and markets*. Cambridge university press Cambridge, 2010, vol. 8.

[3] 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.

[4] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 12, pp. 2724–2743, 2017.

[5] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, 2017.

[6] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018.

[7] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *International Conference on Learning Representations*, 2019.

[8] W. L. Hamilton, "Graph representation learning," *Synthesis Lectures on Artifical Intelligence and Machine Learning*, vol. 14, no. 3, pp. 1–159, 2020.

[9] J. Li, D. Cai, and X. He, "Learning graph-level representation for drug discovery," *arXiv preprint arXiv:1709.03741*, 2017.

[10] W. Fan, Y. Ma, Q. Li, J. Wang, G. Cai, J. Tang, and D. Yin, "A graph neural network framework for social recommendations," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[11] H. Li, X. Wang, Z. Zhang, J. Ma, P. Cui, and W. Zhu, "Intention-aware sequential recommendation with structured intent transition," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[12] M. Zitnik, M. Agrawal, and J. Leskovec, "Modeling polypharmacy side effects with graph convolutional networks," *Bioinformatics*, vol. 34, no. 13, pp. i457–i466, 2018.

[13] T. Sterling and J. J. Irwin, "Zinc 15–ligand discovery for everyone," *Journal of chemical information and modeling*, vol. 55, no. 11, pp. 2324–2337, 2015.

[14] R. S. Bohacek, C. McMartin, and W. C. Guida, "The art and practice of structure-based drug design: a molecular modeling perspective," *Medicinal research reviews*, vol. 16, no. 1, pp. 3–50, 1996.

[15] 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," *Advances in neural information processing systems*, vol. 33, pp. 22 118–22 133, 2020.

[16] Y. Li, B. Qian, X. Zhang, and H. Liu, "Graph neural network-based diagnosis prediction," *Big Data*, vol. 8, no. 5, pp. 379–390, 2020.

[17] X. Han, X. Hu, H. Wu, B. Shen, and J. Wu, "Risk prediction of theft crimes in urban communities: An integrated model of lstm and st-gcn," *IEEE Access*, vol. 8, pp. 217 222–217 230, 2020.

[18] Y. Yang, Z. Wei, Q. Chen, and L. Wu, "Using external knowledge for financial event prediction based on graph neural networks," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 2161–2164.

[19] B. Knyazev, G. W. Taylor, and M. R. Amer, "Understanding attention and generalization in graph neural networks," *Advances in Neural Information Processing Systems*, vol. 32, pp. 4202–4212, 2019.

[20] G. Yehudai, E. Fetaya, E. Meirom, G. Chechik, and H. Maron, "From local structures to size generalization in graph neural networks," in *International Conference on Machine Learning*. PMLR, 2021, pp. 11 975–11 986.

[21] B. Bevilacqua, Y. Zhou, and B. Ribeiro, "Size-invariant graph representations for graph classification extrapolations," in *International Conference on Machine Learning*, 2021, pp. 837–851.

[22] K. Xu, M. Zhang, J. Li, S. S. Du, K.-i. Kawarabayashi, and S. Jegelka, "How neural networks extrapolate: From feedforward to graph neural networks," in *International Conference on Learning Representations*, 2021.

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

[24] L. Tu, G. Lalwani, S. Gella, and H. He, "An empirical study on robustness to spurious correlations using pre-trained language models," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 621–633, 2020.

[25] M. Ren, W. Zeng, B. Yang, and R. Urtasun, "Learning to reweight examples for robust deep learning," in *International Conference on Machine Learning*, 2018, pp. 4334–4343.

[26] K. Kuang, P. Cui, S. Athey, R. Xiong, and B. Li, "Stable prediction across unknown environments," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1617–1626.

[27] P. W. Koh, S. Sagawa, S. M. Xie, M. Zhang, A. Balsubramani, W. Hu, M. Yasunaga, R. L. Phillips, I. Gao, T. Lee *et al.*, "Wilds: A benchmark of in-the-wild distribution shifts," in *International Conference on Machine Learning*, 2021, pp. 5637–5664.

[28] A. Rahimi, B. Recht *et al.*, "Random features for large-scale kernel machines." in *Advances in Neural Information Processing Systems*, vol. 3. Citeseer, 2007, p. 5.

[29] X. Zhang, P. Cui, R. Xu, L. Zhou, Y. He, and Z. Shen, "Deep stable learning for out-of-distribution generalization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 5372–5382.

[30] Z. Li, J.-F. Ton, D. Oglic, and D. Sejdinovic, "Towards a unified analysis of random fourier features," in *International Conference on Machine Learning*. PMLR, 2019, pp. 3905–3914.

[31] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, "Unsupervised feature learning via non-parametric instance discrimination," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3733–3742.

[32] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9729–9738.

[33] J. Lee, I. Lee, and J. Kang, "Self-attention graph pooling," in *International Conference on Machine Learning*, 2019, pp. 3734–3743.

[34] Z. Zhang, J. Bu, M. Ester, J. Zhang, Z. Li, C. Yao, D. Huifen, Z. Yu, and C. Wang, "Hierarchical multi-view graph pooling with structure learning," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[35] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 1025–1035.

[36] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe, "Weisfeiler and leman go neural: Higher-order graph neural networks," in

Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 4602–4609.

[37] W. Yu, X. Lin, J. Liu, J. Ge, W. Ou, and Z. Qin, "Self-propagation graph neural network for recommendation," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[38] J. Li, H. Peng, Y. Cao, Y. Dou, H. Zhang, P. Yu, and L. He, "Higher-order attribute-enhancing heterogeneous graph neural networks," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[39] 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.

[40] J. Gao, J. Gao, X. Ying, M. Lu, and J. Wang, "Higher-order interaction goes neural: A substructure assembling graph attention network for graph classification," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[41] H. Li, X. Wang, Z. Zhang, and W. Zhu, "Out-of-distribution generalization on graphs: A survey," *arXiv preprint arXiv:2202.07987*, 2022.

[42] A. Santoro, F. Hill, D. Barrett, A. Morcos, and T. Lillicrap, "Measuring abstract reasoning in neural networks," in *International Conference on Machine Learning*, 2018, pp. 4477–4486.

[43] D. Saxton, E. Grefenstette, F. Hill, and P. Kohli, "Analysing mathematical reasoning abilities of neural models," in *International Conference on Learning Representations*, 2019.

[44] P. Veličković, R. Ying, M. Padovano, R. Hadsell, and C. Blundell, "Neural execution of graph algorithms," in *International Conference on Learning Representations*, 2020.

[45] A. Loukas, "What graph neural networks cannot learn: depth vs width," in *International Conference on Learning Representations*, 2020.

[46] V. Garg, S. Jegelka, and T. Jaakkola, "Generalization and representational limits of graph neural networks," in *International Conference on Machine Learning*. PMLR, 2020, pp. 3419–3430.

[47] S. Verma and Z.-L. Zhang, "Stability and generalization of graph convolutional neural networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 1539–1548.

[48] M. Cogswell, F. Ahmed, R. Girshick, L. Zitnick, and D. Batra, "Reducing overfitting in deep networks by decorrelating representations," in *International Conference on Learning Representations*, 2016.

[49] S. Gu, Y. Hou, L. Zhang, and Y. Zhang, "Regularizing deep neural networks with an ensemble-based decorrelation method." in *International Joint Conferences on Artificial Intelligence*, 2018, pp. 2177–2183.

[50] D. Arpit, C. Xiong, and R. Socher, "Predicting with high correlation features," *arXiv preprint arXiv:1910.00164*, 2019.

[51] C. J. Song, J. C. Vladescu, K. F. Reeve, C. F. Miguel, and S. L. Breeman, "The influence of correlations between noncritical features and reinforcement on stimulus generalization," *Journal of Applied Behavior Analysis*, vol. 54, no. 1, pp. 346–366, 2021.

[52] M. Hebiri and J. Lederer, "How correlations influence lasso prediction," *IEEE Transactions on Information Theory*, vol. 59, no. 3, pp. 1846–1854, 2012.

[53] P. Rodríguez, J. Gonzalez, G. Cucurull, J. M. Gonfaus, and X. Roca, "Regularizing cnns with locally constrained decorrelations," *arXiv preprint arXiv:1611.01967*, 2016.

[54] Z. Shen, P. Cui, T. Zhang, and K. Kunag, "Stable learning via sample reweighting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 5692–5699.

[55] K. Kuang, R. Xiong, P. Cui, S. Athey, and B. Li, "Stable prediction with model misspecification and agnostic distribution shift," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 4485–4492.

[56] Z. Zhang, Y. Zhang, and Z. Li, "Removing the feature correlation effect of multiplicative noise," *arXiv preprint arXiv:1809.07023*, 2018.

[57] H. Bahng, S. Chun, S. Yun, J. Choo, and S. J. Oh, "Learning de-biased representations with biased representations," in *International Conference on Machine Learning*, 2020, pp. 528–539.

[58] M. Ilse, J. M. Tomczak, C. Louizos, and M. Welling, "Diva: Domain invariant variational autoencoders," in *Medical Imaging with Deep Learning*, 2020, pp. 322–348.

[59] 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.

[60] K. Fukumizu, A. Gretton, X. Sun, and B. Schölkopf, "Kernel measures of conditional dependence." in *Advances in Neural Information Processing Systems*, 2007, pp. 489–496.

[61] E. V. Strobl, K. Zhang, and S. Visweswaran, "Approximate kernel-based conditional independence tests for fast non-parametric causal discovery," *Journal of Causal Inference*, vol. 7, no. 1, 2019.

[62] Y. Yang, Z. Feng, M. Song, and X. Wang, "Factorizable graph convolutional networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 20 286–20 296, 2020.

[63] G. Corso, L. Cavalleri, D. Beaini, P. Liò, and P. Veličković, "Principal neighbourhood aggregation for graph nets," *Advances in Neural Information Processing Systems*, vol. 33, pp. 13 260–13 271, 2020.

[64] H. Gao and S. Ji, "Graph u-nets," in *International Conference on Machine Learning*, 2019, pp. 2083–2092.

[65] P. Erdős, A. Rényi *et al.*, "On the evolution of random graphs," *Publ. Math. Inst. Hung. Acad. Sci*, vol. 5, no. 1, pp. 17–60, 1960.

[66] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[67] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.

[68] A. Shrivastava and P. Li, "A new space for comparing graphs," in *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. IEEE,

2014, pp. 62–71.
[69] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. Vishwanathan, A. J. Smola, and H.-P. Kriegel, "Protein function prediction via graph kernels," *Bioinformatics*, vol. 21, no. suppl_1, pp. i47–i56, 2005.
[70] P. D. Dobson and A. J. Doig, "Distinguishing enzyme structures from non-enzymes without alignments," *Journal of molecular biology*, vol. 330, no. 4, pp. 771–783, 2003.
[71] X. Miao, W. Zhang, Y. Shao, B. Cui, L. Chen, C. Zhang, and J. Jiang, "Lasagne: A multi-layer graph convolutional network framework via node-aware deep architecture," *IEEE Transactions on Knowledge and Data Engineering*, 2021.
[72] Z. Shen, P. Cui, K. Kuang, B. Li, and P. Chen, "Causally regularized learning with agnostic data selection bias," in *Proceedings of the 26th ACM international conference on Multimedia*, 2018, pp. 411–419.
[73] R. Xu, P. Cui, Z. Shen, X. Zhang, and T. Zhang, "Why stable learning works? a theory of covariate shift generalization," *arXiv preprint arXiv:2111.02355*, 2021.
[74] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.
[75] J.-T. Hsieh, B. Liu, D.-A. Huang, L. F. Fei-Fei, and J. C. Niebles, "Learning to decompose and disentangle representations for video prediction," in *Advances in neural information processing systems*, 2018, pp. 517–526.
[76] L. Ma, Q. Sun, S. Georgoulis, L. Van Gool, B. Schiele, and M. Fritz, "Disentangled person image generation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 99–108.
[77] J. Ma, P. Cui, K. Kuang, X. Wang, and W. Zhu, "Disentangled graph convolutional networks," in *International Conference on Machine Learning*, 2019, pp. 4212–4221.
[78] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner, "Understanding disentangling in β-vae," *NeurIPS Workshop on Learning Disentangled Representations*, 2017.

Haoyang Li received his B.E. from the Department of Computer Science and Technology, Tsinghua University in 2018. He is a Ph.D. candidate in the Department of Computer Science and Technology of Tsinghua University. His research interests are mainly in machine learning on graphs and out-of-distribution generalization. He has published several papers in prestigious journals and conferences, e.g., TKDE, KDD, NeurIPS, ICLR, etc.

Xin Wang is currently an Assistant Professor at the Department of Computer Science and Technology, Tsinghua University. He got both of his Ph.D. and B.E degrees in Computer Science and Technology from Zhejiang University, China. He also holds a Ph.D. degree in Computing Science from Simon Fraser University, Canada. His research interests include cross-modal multimedia intelligence and inferable recommendation in social media. He has published several high-quality research papers in top conferences including ICML, MM, KDD, WWW, SIGIR etc. He is the recipient of 2017 China Postdoctoral innovative talents supporting program. He receives the ACM China Rising Star Award in 2020.

Ziwei Zhang received his Ph.D. from the Department of Computer Science and Technology, Tsinghua University, in 2021. He is currently a postdoc researcher in the Department of Computer Science and Technology at Tsinghua University. His research interests focus on machine learning on graphs, including graph neural network (GNN), network embedding (a.k.a. network representation learning), and automated graph machine learning. He has published over 20 papers in prestigious conferences and journals, including KDD, NeurIPS, ICML, AAAI, IJCAI, and TKDE.

Wenwu Zhu is currently a Professor in the Department of Computer Science and Technology at Tsinghua University, the Vice Dean of National Research Center for Information Science and Technology, and the Vice Director of Tsinghua Center for Big Data. Prior to his current post, he was a Senior Researcher and Research Manager at Microsoft Research Asia. He was the Chief Scientist and Director at Intel Research China from 2004 to 2008. He worked at Bell Labs New Jersey as Member of Technical Staff during 1996-1999. He received his Ph.D. degree from New York University in 1996.

His current research interests are in the area of data-driven multimedia networking and Cross-media big data computing. He has published over 350 referred papers, and is inventor or co-inventor of over 50 patents. He received eight Best Paper Awards, including ACM Multimedia 2012 and IEEE Transactions on Circuits and Systems for Video Technology in 2001 and 2019.

He served as EiC for IEEE Transactions on Multimedia (2017-2019). He served in the steering committee for IEEE Transactions on Multimedia (2015-2016) and IEEE Transactions on Mobile Computing (2007-2010), respectively. He serves as General Co-Chair for ACM Multimedia 2018 and ACM CIKM 2019, respectively. He is an AAAS Fellow, IEEE Fellow, SPIE Fellow, and a member of The Academy of Europe (Academia Europaea).