BA-GNN: On Learning Bias-Aware Graph Neural Network

Zhengyu Chen†, Teng Xiao‡, and Kun Kuang†
† College of Computer Science & Technology, Zhejiang University
‡ The Pennsylvania State University
Email: {chenzhengyu,kunkuang}@zju.edu.cn, tengxiao@psu.edu

***Abstract*—Graph Neural Networks (GNNs) show promising results for semi-supervised learning tasks on graphs, thus become favorable comparing with other approaches. However, similar to other machine learning models, GNNs might suffer from the bias issue because of the distribution shift between training and testing node distributions. More importantly, the test node distribution in the graph is generally unknown during mode training in practice. In this paper, we focus on how to address the bias issue on graphs and learn a graph neural network model that is robust to arbitrary unknown distribution shifts. To address this problem, we propose a novel *Bias-Aware Graph Neural Network* (BA-GNN) framework by learning node representations that are invariant across different distributions for invariant prediction. Specifically, our BA-GNN framework contains two interactive parts, one for bias identification and the other for invariant prediction. To learn invariant feature and aggregated representation, our BA-GNN learns multiple biased graph partitions and selects feature, neighbor, and propagation steps for nodes under multiple biased graph partitions. Extensive experiments show that our proposed BA-GNN framework can significantly improve different GNNs backbones such as GCN, GAT and APPNP on different datasets.**

***Index Terms*—Graph Neural Network; Node Classification; Distribution Shift; Bias-Aware**

I. Introduction

Recently, Graph Neural Networks have achieved state-of-the-art performance across various tasks on graphs, such as semi-supervised node classification [1]–[3], link prediction [4], [5] and graph classification [6]. Typically, GNNs exploit message propagation strategy to learn expressive node representations by propagating and aggregating the messages between neighboring nodes. Various message propagation layers have been proposed, including graph convolutional layers (GCN) [1], graph attention layers (GAT) [2] and many others [4], [7]–[10]. Graph Convolutional Networks (GNNs) [1], [2], [7] have achieved great success in many real world applications across different domains, such as recommender system [11], molecule design [12], financial fraud detection [13], traffic prediction [14], and user behavior analysis [15].

Despite the great performance of GNNs, the majority of existing methods assume that the training and testing data are independent and identically distributed (i.e., i.i.d assumption), while for many real-world graphs and applications, the distributions between training and testing data could be different. For instance, in the citation network, the papers with high citations (i.e., high-degree nodes) will be more likely to be labeled and used as training nodes. However, in the testing, we generally have many low-degree nodes that need to be classified. In addition, machine learning (ML)-related papers are more likely to be labeled as training nodes compared to Computer Architecture-related papers. However, a GNN model trained in this citation network may see a vastly different distribution of labels: we have many computer architecture-related papers that need to be classified. Both degree and label shifts between training and testing distributions can significantly degrade model performance as we show later. Figure 1 shows the illustration of degree and label distribution shifts in graph. As shown in the Figure 1, A is the high-degree node with higher influence, which can dominate the training/learning of GNNs. However, the degree of test nodes C and D differ from nodes in training, and most of unlabeled nodes are at the fringes of the graph. The difference of degree distribution can hurt the message-passing mechanism of GNNs. Obviously, GNNs which train on high-degree nodes results in unsatisfying or even poor prediction performance on low-degree nodes. Moreover, as shown in Figure 1 the label of most labeled nodes in training graph is *circle*, which may warp GNNs biased towards *circle* nodes. However, the label

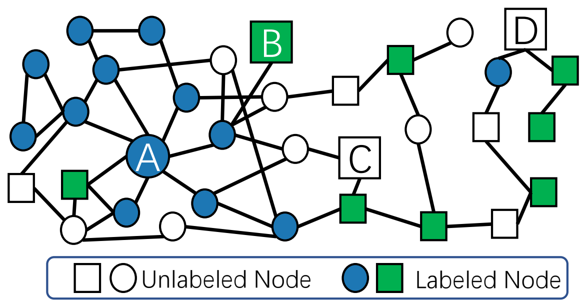
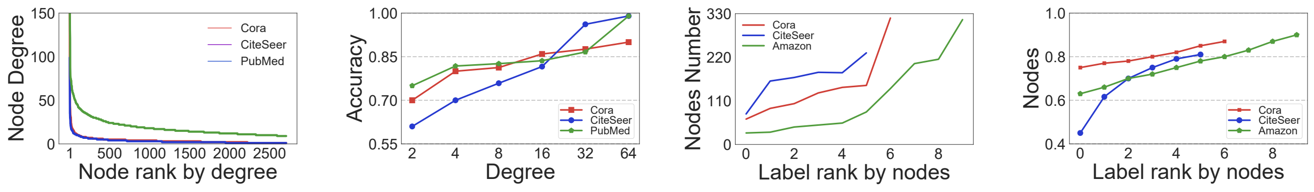


Fig. 1: Schematic diagram of the bias in node classification problem. The shape denotes the label of each node. The shape *circle* is labeled as "machine learning", and the shape *rectangle* is labeled as "computer architecture". The node with high-degree is more likely to be labeled and used as training nodes, however, in the testing, the node with low-degree need to be classified, leading to the distribution (of degrees and label) shifts between training and testing.

The first two authors contributed equally as joint first authorship.
Kun Kuang is the corresponding author, kunkuang@zju.edu.cn.



(a) Degree distribution. (b) Classification performance. (c) Label distribution. (d) Classification performance.

Fig. 2: Empirical investigation of bias in graphs. (a) shows the degree distribution of the Cora, CiteSeer and PubMed datasets. The node degrees are characterized by a long-tailed distribution, where the performance of nodes with different degree varies considerably. (c) shows the label distributions, and the performance of nodes with different labels is shown in (d).

of test nodes C and D is *rectangle* rather than *circle*. Such label-related distribution shift may also degrade the prediction performance on test environments.

Empirical investigation of bias in graph. To further verify and study the distribution shift caused by the bias, we conduct empirical investigation of two common distribution shifts: label-related distribution shift and degree-related distribution shift. Figure 2 (a) shows the degree distribution of the Cora, CiteSeer and PubMed datasets. The node degrees are characterized by a long-tailed distribution, where a significant fraction of the nodes belong to the tail with very low degrees. Generally, the node degrees varies considerably across the graph and are not uniformly distributed, which leads to degree-related distribution shift between training and testing. In addition, Figure 2 (b) shows that the performance of nodes with different degree vary considerably across the graph. The label distributions of datasets are shown in Figure 2 (c), which are not uniformly distributed. GNNs rely on message-passing mechanism, and aggregate the information from neighbors to learn representations. Thus, the nodes where their label has fewer nodes receive less information during the aggregation, which leads to poor performance as shown in Figure 2 (d).

More importantly, the test distribution is always unknown during GNNs optimizing on training graph data, where the unknown distribution shift might be caused by node labels, node degrees, or both. Generally, the unknown distribution shift between training and testing would render traditional GNNs over-optimized on the labeled training samples and render their predictions error prone on test samples, resulting in variant predictions. Therefore, learning GNNs that are resilient to distribution shifts and able to make invariant predictions on graphs will be important for real-world applications.

In this paper, we focus on the bias issue on graph and study a novel problem of learning de-biased graph neural networks for unknown testing distributions. To address this problem, we are still facing the following challenges: 1) *How to overcome distribution shifts caused by bias.* Previous works [16]–[22] only focus on graph-level task or degree bias in node-level task, which could not address other bias such as label bias in node-level task. 2) *How to overcome bias problem in unknown test environment.* The testing distribution is always unknown during GNNs optimizing on training graph data, previous works [16]–[22] ignore the bias in unknown test environments. 3) *How to overcome bias problem in graph data.* Previous methods [23]–[36] are for bias problem with independent and identically distributed (i.i.d) data, however, graph-structured data is non i.i.d, and the properties of graph-structured data are not explicitly utilized in these methods.

In an attempt to address these challenges, in this work, we propose a novel *Bias-Aware Graph Neural Network* (BA-GNN) framework that aims to learn invariant graph representations for robust prediction across unknown testing distributions. Our BA-GNN framework contains two interactive parts, the frontend $\mathcal{M}_B$ for bias identification and the backend $\mathcal{M}_I$ for invariant prediction. To learn invariant feature and aggregated representation, $\mathcal{M}_B$ learns multiple biased graph partitions, and $\mathcal{M}_I$ selects feature, neighbor and propagation step for nodes under multiple biased graph partitions.

We compare our BA-GNN framework with a bunch of generic SOTA GNNs and methods that are specifically designed for mitigating selection biases on various public graph benchmarks. We concern both traditional task-specific evaluation metrics and protocols that are especially designed for invariant learning. Extensive experimental results demonstrate the capability of our framework on learning GNNs that makes invariant predictions on graphs with unknown testing distribution. In summary, the contributions of this paper are:

- We study the problem of learning GCNs with bias for invariant prediction across unknown test environments, which is less explored in the literature.
- We propose a novel framework *Bias-Aware Graph Neural Network* (BA-GNN) for GNNs, which learns invariant aggregated presentation for each node, and make invariant prediction on various unknown test environments.
- Extensive experiments show that our proposed BA-GNN framework can significantly improve different GNNs backbones such as GCN, GAT and APPNP on different datasets and settings.

The remainder of this paper is organized as follows. We review about the related work in Section 2. We present problem formulation for the node classification problem with bias on unknown test environments in Section 3. It is followed by the elaboration of the proposed BA-GNN approach in Section 4. In Section 5, we present extensive experimental studies on different public graph benchmarks. Finally, the conclusion of this paper is presented in Section 6.

II. Related Works

A. Graph Neural Networks

GNNs have achieved great success for modeling graph structured data. Generally, GNNs can be categorized into two categories, i.e., spectral-based and spatial-based. Spectral-based GNNs define graph convolution based on spectral graph theory [3], [10], [37]. GCN [1] further simplifies graph convolutions by stacking layers of first-order Chebyshev polynomial filters together with some approximations. Spatial-based methods directly define updating rules in the spatial space. For instance, GAT [2] introduces the self-attention strategy into aggregation to assign different importance scores of neighborhoods. With the similar intuition, GraphSAGE [4] extends prior works in the inductive setting. There is a lot of spatial-based methods [2], [21], [38]–[40] are proposed to capture different neighborhood information.

The spectral based GNNs usually require to compute the Laplacian eigenvectors or the approximated eigenvalues as suggested by spectral theory, and these methods are inefficient on large scale graph. Different from the spectral based ones, the spatial-based GNNs [41]–[43] attempt to directly capture the spatial topological information and use the mini-batch training schema. For example, DCNN [41] combines graph convolutional operator with the diffusion process, and Veličković et al. propose the graph attention network [42] with the self-attention mechanism on the neighbors of nodes and assign different weights during the aggregation process.

B. Bias in Machine Learning

Recently, many methods are proposed to address bias caused by distribution shift for general machine learning problems. There are mainly three branches of methods for the selection bias caused by distribution shift, namely domain adaptation [23]–[25], distributionally robust optimization (DRO) [26]–[31] and invariant learning [32]–[35].

Domain adaptation methods aim to reduce bias by learning domain-invariant representations, which is learned by minimizing a certain discrepancy between distributions of source and target features extracted by a shared representation learner [23]–[25]. [23] put forward the domain adversarial neural network (DANN). A domain discriminator is trained to distinguish source features from target features and a feature extractor to confuse the discriminator. Since then, a series of works have appeared and achieved significantly better performance. [24] proposed an architecture that employed asymmetric encodings for target and source data. [44] presented a principled framework that conducted the adversarial adaptation models using conditional information. [25], [45] unified pixel-level and feature-level adversarial learning for domain adaptation. [46] considered the classifiers instead of features and designed an original adversarial learning method by maximizing the classifier discrepancy.

DRO methods propose to optimize the worst-case risk within an uncertainty set, which lies around the observed training distribution and characterizes the potential testing distributions [30], [31]. However, to better capture the testing distribution, the uncertainty set should be pretty large in many real-world scenarios, which also results in the over-pessimism problem of DRO methods [30], [31].

Realizing the difficulty of solving selection bias caused by distribution shift problem without any prior knowledge or structural assumptions, invariant learning methods assume the existence of causally invariant relationships between some predictors $\Phi(X)$ and the target Y [32], [35], [36]. [32] and [33] propose to learning an invariant representation through multiple training environments. [34] also proposes to select features whose predictive relationship with the target stays invariant across environments. However, their effectiveness relies on the quality of the given multiple training environments, and the role of environments remains vague theoretically. Recently, [35] improves [32] by relaxing its requirements for multiple environments. Specifically, [35] proposes a two-stage method, which firstly infers the environment division with a pre-provided biased model, and then performs invariant learning on the inferred environments.

As summary, these methods mentioned above are generally adopted from general machine learning tasks, where the data is independent and identically distributed (i.i.d). However, graph-structured data is non i.i.d, and the properties of graph-structured data are not explicitly utilized in these methods.

Recent works explore GNN's extrapolation ability to address bias caused by distribution shift. [16] proposes to learn a static adjacency matrix for a given graph and expects that the learned adjacency matrix captures general relational patterns that are free from selection biases. [17] suggests that encoding appropriate non-linearity in architecture and features can help extrapolation. [18] show how subgraph densities can be used to build size-invariant graph representations. [19] propose a Self-Supervised Learning (SSL) task aimed at learning representations of local structures to overcome the size-generalization problem. However, they concentrate on *graph-level tasks* (e.g., graph classification), where each input instance is a graph (usually with less than 100 nodes) and one dataset contains massive graphs for training and testing.

Recent works study the degree bias in node-level task. GNM [20] confronts a related problem named non-ignorable nonresponse, which indicates that the unlabeled nodes are missing not at random (MNAR). DEMO-Net [21] explicitly capture the graph topology integrated with node attributes. SL-DSGCN [22] mitigate the degree-related biases of GCNs from model and data aspects. However, these works only focus on degree bias, and ignore other bias. Moreover, these methods could not address bias in unknown environments.

III. Problem Formulation

In this section, we present our problem formulation for the node classification with bias on unknown test environments.

An input graph $G = (A, X, Y)$ contains two-folds information: an adjacency matrix A and node features X. Each node in the graph has a label, which is denoted as Y. Following [28], we define a graph environment as the joint distribution P_{XAY} on $X * A * Y$ and use $\mathcal{E}$ denote the set of all environments. For each environment $e \in \mathcal{E}$, we have a graph dataset $G^e = (X^e,\ A^e, Y^e)$, where $X^e \in \mathcal{X}$ are node features, $A^e \in \mathcal{A}$ is the adjacency matrix, and $Y^e \in \mathcal{Y}$ is the response variable (e.g., node labels in the node classification problem). The joint distribution of features and outcomes on (X, A, Y) can vary across environments, i.e., $P^e_{XAY} \neq P^{e'}_{XAY}$ for $e, e' \in \mathcal{E}$, and $e \neq e'$. In this paper, we aim to learn node representations based on which we can make invariant predictions across environments with various unknown biases.

Node classification problem with bias on unknown test environments. Given a training graph $\mathcal{G}_{\text{train}} = \{A_{\text{train}}, X_{\text{train}}, Y_{\text{train}}\}$, the task is to learn a GNN $g_\theta(\cdot)$ with parameter θ to precisely predict the label of nodes on different unknown test graphs $\{\mathcal{G}^1_{\text{test}}, \mathcal{G}^2_{\text{test}}, \cdots, \mathcal{G}^e_{\text{test}}\}$, where $\mathcal{G}^e_{\text{test}} = \{A^e_{\text{test}}, X^e_{\text{test}}, Y^e_{\text{test}}\}$.

IV. Methods

A. Graph Neural Networks

It has been observed that a broad class of graph neural network (GNN) architectures followed the 1-dimensional Weisfeiler-Lehman (WL) graph isomorphism test [47]. From the perspective of WL isomorphism test, they mainly consist of the following crucial steps at each iteration of feature aggregation:

- Feature initialization (label initialization): The node features are initialized by original attribute vectors.
- Neighborhood detection (multiset-label determination): It decides the local neighborhood in which node gathers the information from neighbors. More specifically, a seed followed by its neighbors generates a subtree pattern.
- Neighbors sorting (multiset-label sorting): The neighbors are sorted in the ascending or descending order of degree values. The subtrees with permutation order of neighbors are recognized as the same one.
- Feature aggregation (label compression): The node feature is updated by compressing the feature vectors of the aggregated neighbors including itself.
- Graph-level pooling (graph representation): It summarizes all the node features to form a global graph representation.

We would like to point out that graph neural networks would learn the node or graph representation using continuous node attributes, whereas WL algorithms update the node attributes by directly compressing the augmented discrete attributes.

Taking 1-hop neighborhood $N(v) = \{u|(v, u) \in E\}$ into consideration at each iteration, the following node-level graph neural network variants have the same feature initialization and neighborhood detection on learning node representation. And when element-wise average or max operations are used for feature aggregation, graph neural networks would be invariant to the order of neighbors.

Graph Convolutional Network (GCN) [1]:

$$h^k_v = \sigma\Big(\sum\nolimits_{u\in\{v\}\cup N(v)} \widehat{a}_{vu} W^k h^{k-1}_u\Big) \tag{1}$$

where $\widehat{A} = (\widehat{a}_{vu}) \in \mathbb{R}^{n\times n}$ is the re-normalization of the adjacency matrix A with added self-loops, and W^k is the trainable matrix at k^{th} layer. It is essentially a weighted feature aggregation from node neighborhood.

B. Bias-Aware Graph Neural Network

Despite the great performance of GNNs, some distribution-specific patterns might warp the GNN biased towards a globally sub-optimal solution since mostly yield distribution shift from the training graph distribution and the testing data distribution in real-world applications.

Without any prior knowledge or structural assumptions, it is impossible to figure out the *node classification problem with bias on unknown environments*, since one cannot characterize the unseen latent environments in $\mathrm{supp}(\mathcal{E})$.

In graph data, due to the neighborhood aggregation process, each neighbor node contributing to the final aggregated representation can be viewed as a property of the root node. Thus, we should consider all invariant properties of the target node, such as *feature*, *edge* and *propagation step*. In invariant graph learning, we have assumption based on a commonly used assumption in invariant learning literature [32]–[36]:

Assumption 4.1: There exists random variable $\Phi^*(X, A)$ such that the following properties hold:

a. Invariance property: for all $e, e' \in \mathrm{supp}(\mathcal{E})$, we have $P^e(Y|\Phi^*(X, A)) = P^{e'}(Y|\Phi^*(X, A))$ holds.

b. Sufficiency property: $Y = f(\Phi^*) + \epsilon,\ \epsilon \perp X$.

However, as shown in the Figure 1, there are degree and label biases between training and testing. The environments need to be subtly uncovered, as indicated by Theorem 4.1, not all environments are helpful to tighten the invariance set.

Theorem 4.1: Given set of environments $\mathrm{supp}(\hat{\mathcal{E}})$, denote the corresponding invariance set $\mathcal{I}_{\hat{\mathcal{E}}}$ and the corresponding maximal invariant predictor $\hat{\Phi}$. For one newly-added environment e_{new} with distribution $P^{new}(X, A, Y)$, if $P^{new}(Y|\hat{\Phi}) = P^e(Y|\hat{\Phi})$ for $e \in \mathrm{supp}(\hat{\mathcal{E}})$, the invariance set constrained by $\mathrm{supp}(\hat{\mathcal{E}}) \cup \{e_{new}\}$ is equal to $\mathcal{I}_{\hat{\mathcal{E}}}$.

Besides Assumption 4.1, we make another assumption on the existence of bias in training data as:

Assumption 4.2: $\mathrm{Bias-Aware\ Assumption.}$
For random variable pair (X, A, Φ^*) and Φ^* satisfying Assumption 4.1, using functional representation lemma [48], there exists random variable Ψ^* such that $X = X(\Phi^*, \Psi^*)$, then we assume $P^e(Y|\Psi^*)$ can arbitrary change across environments $e \in \mathrm{supp}(\mathcal{E})$.

Theorem 4.1 together with Assumption 4.2 indicate that, to better constrain $\mathcal{I}_{\mathcal{E}_{tr}}$, the effective way is to generate environments with varying $P(Y|\Psi^*(X, A))$ that can exclude variant features from $\mathcal{I}_{\mathcal{E}_{tr}}$. Under this setting, we encounter the circular dependency: first we need variant Ψ^* to generate multiple environments $\mathcal{E}_{tr}$; then we need $\mathcal{E}_{tr}$ to learned invariant Φ^* as well as variant Ψ^*. Furthermore, there exists positive feedback between these two steps. When acquiring $\mathcal{E}_{tr}$ with tighter $\mathcal{I}_{\mathcal{E}_{tr}}$, more invariant predictor $\Phi(X, A)$ can be found, which will further bring a clearer picture of variant parts, and therefore promote the generation of $\mathcal{E}_{tr}$.

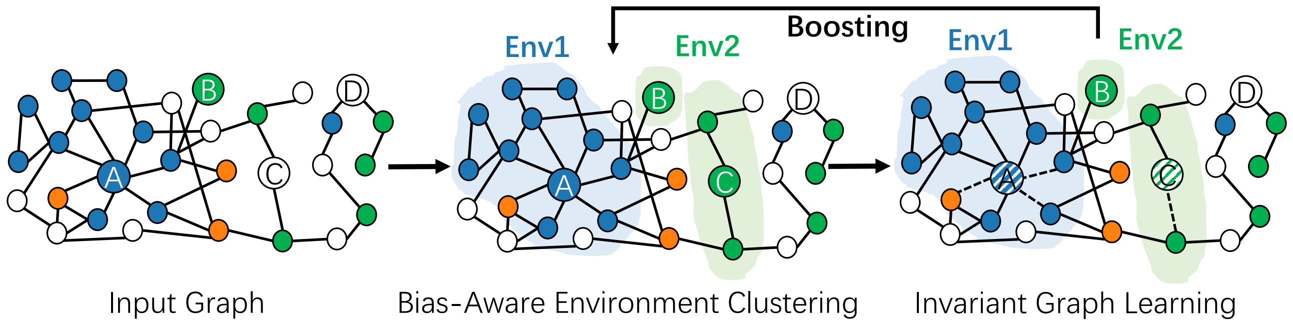


Fig. 3: The framework of BA-GNN. BA-GNN aims to learn invariant graph representations. BA-GNN contains two interactive parts: *Bias-Aware Environment Clustering* for bias identification and *Invariant Graph Learning* for learning invariant aggregated representation.

With this notion, in this work, we propose a novel *Bias-Aware Graph Neural Network* (BA-GNN) framework that aims to learn a de-biased graph representation. Our BA-GNN framework contains two interactive parts, the frontend $\mathcal{M}_B$ for bias identification and the backend $\mathcal{M}_I$ for invariant graph learning. BA-GNN leverages the mutual promotion between the two steps and conduct joint optimization. The general framework is shown in Figure 3.

Given the graph data, it starts with the bias identification module $\mathcal{M}_B$ leveraging the learned variant representation $\Psi(X)$ and variant adjacency matrix $\Psi(A)$ to generate bias environments $\mathcal{E}_{learn}$. Then the learned environments are used by invariant graph learning module $\mathcal{M}_I$ to learn the $\Phi(X, A)$ as well as the invariant graph learning model GNN($\Phi(X, A)$). After that, we derive the variant $\Psi(X, A)$ to further boost the module $\mathcal{M}_B$, which is supported by Theorem 4.1. As for the 'convert' step, we adopt feature selection in this work, through which more variant feature Ψ can be attained when more invariant feature Φ is learned. Specifically, the invariant predictor of feature $\Phi(X)$ is generated as $\Phi(X) = M_x \odot X$, and the variant part $\Psi(X) = (1 - M_x) \odot X$ correspondingly, where $M \in \{0, 1\}^d$ is the binary invariant feature selection mask. The invariant predictor of adjacency $\Phi(A)$ is similar to $\Phi(X)$, where $\Phi(A) = M_a \odot A$, and the variant part $\Psi(A) = (1 - M_a) \odot A$.

For instance, the feature information is more important for final aggregated representation of node B compared with node A, the reason is that node B is low-degree nodes, low-degree nodes only have few neighbors, which receive very limited information from neighborhoods. Thus, the aggregated representation should have more information of feature for the node B. Moreover, the edges that bring the noise to the representation should be masked, as shown in Figure 4 (b), the dash line are edges that are masked. In addition, the propagation step influences the features of nodes locally along the edges of the graph. In Figure 4 (c), we can obviously observe that the optimal propagation step for the node A is 2, however, the optimal step for the node B is 1 since more steps will bring the noise to the representation of it. Thus, the parameter of GCNs in higher layers should be masked for the node B, and should not be masked for the node A. Note that we use the soft selection which is more flexible and general in our algorithm with $M \in [0, 1]^d$. The whole framework is jointly optimized, so that the mutual promotion between bias identification and invariant learning can be fully leveraged.

C. Bias-Aware Environment Clustering

The bias identification module $\mathcal{M}_B$ takes a single graph as input, and outputs a multi-environment graph partition for invariant graph learning. We implement it with a clustering algorithm. As indicated in Theorem 4.1, the more diverse $P(Y|\Psi(X, A))$ for our generated environments, the better the invariance set $\mathcal{I}$ is. Therefore, we cluster the nodes according to the relationship between $\Psi(X, A)$ and Y, for which we use $P(Y|\Psi(X, A))$ as the cluster centre. Note that $\Psi(X, A)$ is initialized as $\Psi(X, A) = (X, A)$ in our joint optimization.

To learn cluster centre $P(Y|\Psi(X, A))$, we assume the j-th cluster centre $P_{\theta_j}(Y|\Psi(X, A))$ parameterized by GCN f_{θ_j} as:

$$h_j(\Psi, Y) = P_{\theta_j}(Y \mid \Psi) = \left\| Y - f_{\theta_j}\left(\Psi(X_j, A)\right) \right\|_2^2, \quad (2)$$

where Ψ means the learned variant part $\Psi(X, A)$. For the given $N = \sum_{e \in \mathcal{E}_{tr}} n_e$ nodes $(X, A) == \{\psi_i(x_i, a_{i,:}), y_i\}_{i=1}^N$, the empirical distribution is modeled as $\hat{P}_N = \frac{1}{N}\sum_{i=1}^N \delta_i(\Psi(X, A), Y)$ where

$$\delta_i(\Psi, Y) = \begin{cases} 1, & \text{if } \Psi(X, A) = \psi_i(x_i, a_{i,:}) \text{and } Y = y_i \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

The target of our bias clustering is to find a distribution in $\mathcal{Q} = \{Q | Q = \sum_{j \in [K]} q_j h_j(\Psi(X, A), Y), \mathbf{q} \in \Delta_K\}$ to fit the empirical distribution best, where Δ_K means K-dimension simplex.. Therefore, the objective function of our bias clustering is:

$$\min_{Q \in \mathcal{Q}} D_{KL}(\hat{P}_N \| Q) \quad (4)$$

The above objective can be further simplified to:

$$\min_{\Theta, \mathbf{q}} \left\{ \mathcal{L}_c = -\frac{1}{N} \sum_{i=1}^N \log \left[\sum_{j=1}^K q_j h_j(\psi_i(x_i, a_{i,:}), y_i) \right] \right\} \quad (5)$$

As for optimization, we use EM algorithm to optimize the centre parameter Θ and the mixture weight $\mathbf{q}$. After

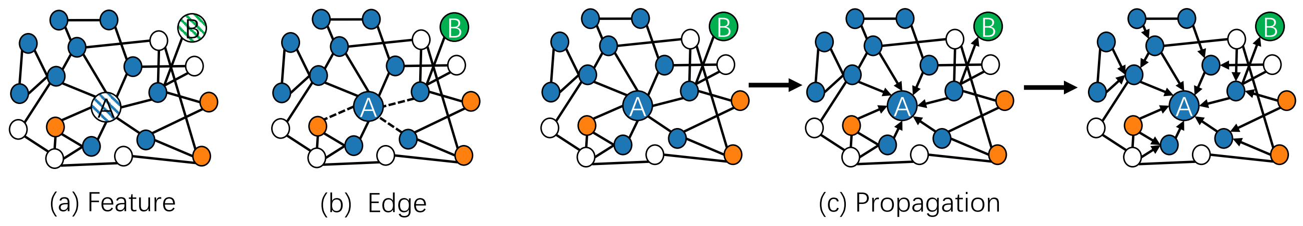


Fig. 4: Illustrations of our Invariant Graph Learning. (a) Learn most informative feature which is invariant across environments. (b) Mask edges that bring variant and noise information to final aggregated representation. (c) Learn optimal propagation step for nodes across different environments.

optimizing equation 5, for building $\mathcal{E}_{tr}$, we assign each node to environment $e_j \in \mathcal{E}_{tr}$ with probability:

$$P(e_j|\Psi(X,A),Y) = q_j h_j(\Psi(X,A),Y) / \left(\sum_{i=1}^{K} q_i h_i(\Psi(X,A),Y) \right) \tag{6}$$

In this way, $\mathcal{E}_{tr}$ is generated by $\mathcal{M}_B$.

D. Invariant Graph Learning

Here we introduce our invariant graph learning module $\mathcal{M}_I$, which takes multiple environments training graph partition $G = \{G^e\}_{e \in supp(\mathcal{E}_{tr})}$ as input, and outputs the corresponding invariant predictor GNN_θ, the invariant feature mask M_x, invariant adjacency M_a and the propagation mask M_p. We combine feature selection, neighborhood selection and propagation selection with invariant learning under bias environments, which can learned invariant aggregated features with invariant correlations of the label across $\mathcal{E}_{tr}$. Specifically, the former module can select most informative features, neighbors and propagation step with respect to the loss function and latter module ensures the aggregated representation is invariant.Their combination ensures $\mathcal{M}_I$ to select the most informative invariant features, neighbors and propagation steps.

Given one graph partition G^e, the neighborhood aggregation in most GNNs with our invariant mask (M_x, M_a, M_p) is:

$$\mathrm{h}'_i = M_{p,i}\sigma \left(\sum_{j \in \mathcal{N}_i} M_{a,ij} \ \mathrm{W}(M_{x,j} \odot \mathrm{h}_j) \right) \tag{7}$$

where h_i denotes the updated representation of the target node i , W denotes the weight matrix of the linear transformation, σ denotes the nonlinear activation function, $\mathcal{N}_i$ denotes the indices set of node i 's neighbors, and $M_{x,j}, M_{a,ij}$ are invariant feature and neighbor selection mask, respectively. $M_{p,i}$ is the propagation mask, where $M_{p,i} = 0$ means exiting the propagation process, and $M_{p,i} = 1$ means proceeding the propagation process.

For invariant learning, we follow the variance penalty regularizer proposed in [33] and simplify it in feature, neighbor and propagation step selection scenarios. The objective function of $\mathcal{M}_I$ with $(M_x, M_a, M_p) \in \{0,1\}^d$ is:

$$\mathcal{L}^e(M \odot (X,A), Y; \theta) = \mathbb{E}_{P^e}[\ell(GNN(X^e, A^e, M_x, M_a, M_p; \theta), Y^e)] \tag{8}$$

$$\mathcal{L}_p(M \odot (X,A), Y; \theta) = \mathbb{E}_{\mathcal{E}_{tr}}[\mathcal{L}^e] + \lambda \mathrm{trace}(\mathrm{Var}_{\mathcal{E}_{tr}}(\nabla_\theta \mathcal{L}^e)) \tag{9}$$

However, as the optimization of hard feature, neighbor, and propagation step selection with binary mask M suffers from high variance, we use the soft selection with gates taking continuous value in $[0,1]$. Specifically, following [49], we approximate each element of $M = [m_1, \ldots, m_d]^T$ to clipped Gaussian random variable parameterized by $\mu = [\mu_1, \ldots, \mu_d]^T$ as

$$m_i = \max\{0, \min\{1, \mu_i + \epsilon\}\} \tag{10}$$

where ϵ is drawn from $\mathcal{N}(0, \sigma^2)$. With this approximation, the objective function with soft selection can be written as:

$$\begin{aligned}\mathcal{L}^e(\theta, \mu) = \mathbb{E}_{P^e}\mathbb{E}_M\,[\ell(GNN(X^e, A^e, M_x, M_a, M_p; \theta), Y^e) \\ +\alpha(\|M_a\|_0 + \|M_x\|_0 + \|M_p\|_0)]\end{aligned} \tag{11}$$

Under the approximation in Equation 10, $\|M_x\|_0$ is simply $\sum_{i \in [d]} P(m_i > 0)$ and can be calculated as $\|M_x\|_0 = \sum_{i \in [d]} \mathrm{CDF}(\mu_{i,x}/\sigma)$, where CDF is the standard Gaussian CDF. Similarly, we have $\|M_a\|_0 = \sum_{i \in [A]} \mathrm{CDF}(\mu_{i,a}/\sigma)$ and $\|M_p\|_0 = \sum_{i \in [P]} \mathrm{CDF}(\mu_{i,p}/\sigma)$, where P is the maximal propagation step given by layer number of GNNs. $\mu = \{\mu_x, \mu_a, \mu_p\}$. We formulate our objective as risk minimization problem:

$$\min_{\theta,\mu} \mathcal{L}_p(\theta; \mu) = \mathbb{E}_{\mathcal{E}_{tr}}[\mathcal{L}^e(\theta, \mu_x, \mu_a, \mu_p)] + \lambda \mathrm{trace}(\mathrm{Var}_{\mathcal{E}_{tr}}(\nabla_\theta \mathcal{L}^e)) \tag{12}$$

where

$$\mathcal{L}^e(\theta, \mu) = \mathbb{E}_{P^e}\mathbb{E}_M\,[\ell(GNN(X^e, A^e, M_x, M_a, M_p; \theta), Y^e) \tag{13}$$

$$+\alpha \sum_{i \in [P]} \mathrm{CDF}(\mu_p/\sigma) + \alpha \sum_{i \in [A]} \mathrm{CDF}(\mu_a \sigma) + \alpha \sum_{i \in [d]} \mathrm{CDF}(\mu_i/\sigma) \Bigg]$$

The whole algorithm of our proposed BA-GNN framework is detailed in Algorithm 1.

V. Theoretical Analysis

In this section, we theoretically analyze our proposed Bias-Aware Graph Neural Network (BA-GNN) method.

Proof of Theorem 4.1 We denote the invariance set with regard to $\mathrm{supp}(\hat{\mathcal{E}} \cup \{e_{new}\})$ as $\mathcal{I}_{new}$, we can easily prove that $\forall S \in \mathcal{I}_{\hat{\mathcal{E}}}$, we have $S \in \mathcal{I}_{new}$, since the newly-added environment cannot exclude any variables from the original invariance set.

Justification of $\mathcal{M}_I$ We prove that the invariant graph learning module $\mathcal{M}_I$ can learn the maximal invariant predictor

Algorithm 1 Bias-Aware Graph Neural Network (BA-GNN)

```
1: Require: Require: Graph data $G = (A, X)$ and label $Y$.
2: while Not converged or maximum epochs not reached do
3:     Obtain multi-environment graph partition $G^j = (A^j, X^j)$ via bias-aware environment clustering module $\mathcal{M}_B$.
4:     for all $e = 1$ to $|\mathcal{E}_{tr}|$ do
5:         Computing loss as in Equation 13.
6:     end for
7:     Optimize $\theta, \mu$ to minimize $\mathcal{L}_p$ as in Equation 12.
8: end while
```

$\Phi(X, A)$ with respect to the corresponding invariance set $\mathcal{I}_{\mathcal{E}_{tr}}$ given training environments $\mathcal{E}_{tr}$.

Theorem 5.1: Given $\mathcal{E}_{tr}$, the learned $\Phi(X, A) = M \odot (X, A)$ is the maximal invariant predictor of $\mathcal{I}_{\mathcal{E}_{tr}}$, where $M = \{M_x, M_a, M_p\}$.

Justification of the Positive Feedback The mechanism for $\mathcal{M}_B$ and $\mathcal{M}_I$ to mutually promote each other is the core of our BA-GNN framework. Such *positive feedback* is theoretically justified in this paper. In Assumption 4.1, we assume that the invariance and sufficiency properties of the invariant features Φ^* and the relationship between variance part Ψ^* and Y can arbitrarily change. With respect to Φ^* and Ψ^*, we have a more specific assumption on the bias-aware across environments.

Assumption 5.1: Assume the training graph data is made up of different biased data sources: $P_{tr} = \sum_{e \in \text{supp}(\mathcal{E}_{tr})} w_e P^e$. For any $e_i, e_j \in \mathcal{E}_{tr}, e_i \neq e_j$, we assume

$$I^c_{i,j}(Y; \Phi^* | \Psi^*) \geq \max(I_i(Y; \Phi^* | \Psi^*), I_j(Y; \Phi^* | \Psi^*)) \tag{14}$$

where Φ^* is invariant part and Ψ^* the variant. I_i represents mutual information in P^{e_i} and $I^c_{i,j}$ represents the cross mutual information between P^{e_i} and P^{e_j} takes the form of $I^c_{i,j}(Y; \Phi | \Psi) = H^c_{i,j}[Y|\Psi] - H^c_{i,j}[Y|\Phi, \Psi]$ and $H^c_{i,j}[Y] = -\int p^{e_i}(y) \log p^{e_j}(y) dy$.

This assumption could be demonstrated intuitively. Firstly, the mutual information $I_i(Y; \Phi^*) = H_i[Y] - H_i[Y|\Phi^*]$ can be viewed as the error reduction if we utilize Φ^* to predict Y instead of nothing. Then the cross mutual information $I_{i,j}(Y; \Phi^*)$ can be viewed as the error reduction if we utilize the predictor learned on Φ^* in environment e_j to predict in environment e_i, rather than predict by nothing. As a result, the R.H.S in Equation 14 measures that, how much prediction error can be decreased in environment e_i, if we further add Φ^* for prediction rather than use only Ψ^*. And the L.H.S measures that how much prediction error can be decreased when utilizing predictors trained in e_i to predict in e_j, if we also add Φ^* for prediction rather than use only Ψ^*. Intuitively, Assumption 5.1 assumes that invariant feature Φ^* provides more information for predicting Y across environments than in one single environment, and that the information provided by Ψ^* shrinks significantly across environments, which indicates that the relationship between variant part Ψ^* and Y varies across environments.

Based on this assumption, we first prove that the cluster centers are pulled apart as invariant feature is removed from clustering.

Theorem 5.2: For $e_i, e_j \in \text{supp}(\mathcal{E}_{tr})$, assume that $X = [\Phi^*, \Psi^*]^T$ satisfying Assumption 4.1, where Φ^* is invariant and Ψ^* variant. Then under Assumption 5.1, we have $\mathrm{D_{KL}}(P^{e_i}(Y|X, A) \| P^{e_j}(Y|X, A)) \leq \mathrm{D_{KL}}(P^{e_i}(Y|\Psi^*) \| P^{e_j}(Y|\Psi^*))$

Theorem 5.2 indicates that when employing variant part Ψ^* the distance between cluster centers is larger, making it is more likely to obtain the desired biased environments, which explains the reason we utilize learned variant part $\Psi(X, A)$ for clustering. Finally, we provide the theorem for optimality guarantee for our framework.

Theorem 5.3: Under Assumption 4.1 and 5.1, for the proposed $\mathcal{M}_B$ and $\mathcal{M}_I$, we have the following conclusions: 1. Given environments $\mathcal{E}_{tr}$ such that $\mathcal{I}_{\mathcal{E}} = \mathcal{I}_{\mathcal{E}_{tr}}$, the learned $\Phi(X, A)$ by $\mathcal{M}_I$ is the maximal invariant predictor of $\mathcal{I}_{\mathcal{E}}$.

Given the maximal invariant predictor Φ^* of $\mathcal{I}_{\mathcal{E}}$, assume the training graph data is made up of data from all environments in $\text{supp}(\mathcal{E})$, there exist one split that achieves the minimum of the objective function and meanwhile the invariance set regularized is equal to $\mathcal{I}_{\mathcal{E}}$.

Intuitively, Theorem 5.3 proves that given one of the $\mathcal{M}_B$ and $\mathcal{M}_I$ optimal, the other is optimal, which validates the existence of the global optimal point of our BA-GNN framework.

VI. Experiments

In this section, we conduct experiments on graph benchmarks to evaluate the effectiveness of the proposed frameworks with comparison to state-of-the-art GNNs. Specifically, we aim to answer the following questions:

- **(RQ 1)** How effective is the proposed BA-GNN framework for the node classification task on different graphs and different backbones?
- **(RQ 2)** Could the proposed BA-GNN alleviate bias issue?
- **(RQ 3)** Could the proposed framework learn to identify different biases?
- **(RQ 4)** How efficiency is the proposed BA-GNN framework with comparison to state-of-the-art GNNs?

A. Datasets

We conduct experiments on 7 datasets based on citation, co-authorship, or co-purchase graphs for semi-supervised node classification tasks; those are Cora [50], CiteSeer [50], PubMed [50], Coauthor CS [51], Coauthor Physics [51], Amazon Computers [51], and Amazon Photo [51]. The statistics of datasets are summarized in the Table I.

Citation datasets. Cora, CiteSeer and PubMed [50] are representative citation network datasets where nodes and edges denote documents and their citation relationships, respectively. Node features are formed by bay-of-words representations for documents. Each node has a label indicating what field the corresponding document belongs to.

Co-authorship datasets. Coauthor CS and Coauthor Physics [51] are co-authorship graphs datasets. Nodes denote

TABLE I: Statistics of datasets. The edge density is computed by $\frac{2m}{n^2}$. The standard fixed training/validation/testing split is widely used in [1]–[3].

Dataset	#Classes	#Nodes	#Edges	Edge Density	#Features	#Training Nodes	#Validation Nodes	#Test Nodes
Cora	7	2708	5278	0.0014	1433	20 per class	500	1000
CiteSeer	6	3327	4552	0.0008	3703	20 per class	500	1000
PubMed	3	19717	44324	0.0002	500	20 per class	500	1000
Coauthor CS	15	18333	81894	0.0005	6805	20 per class	30 per class	Rest nodes
Coauthor Physics	5	34493	247962	0.0004	8415	20 per class	30 per class	Rest nodes
Amazon Computers	10	13381	245778	0.0027	767	20 per class	30 per class	Rest nodes
Amazon Photo	8	7487	119043	0.0042	745	20 per class	30 per class	Rest nodes

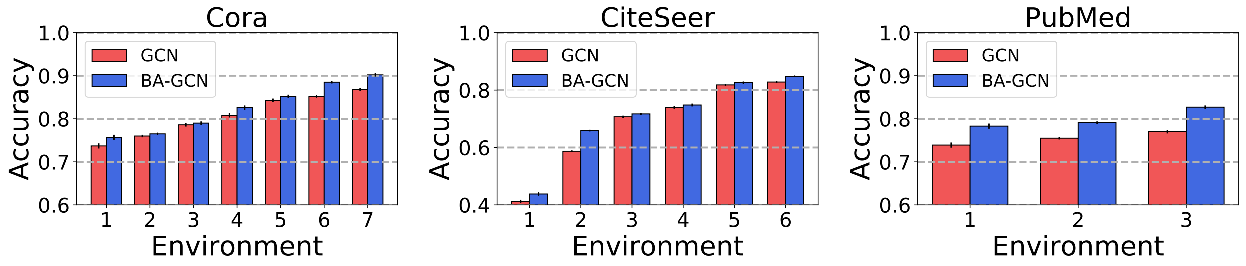


Fig. 5: Results of **GCN backbone** under **label-related distribution shift** for the task of semi-supervised node classification. Comparing with GCN method, our BA-GCN method (by applying our BA-GNN framework on GCN backbone) improves the accuracy of node classification across different label biased environments.

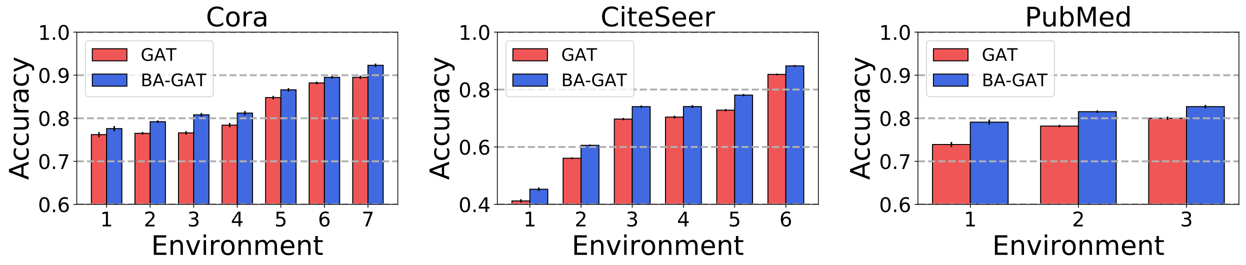


Fig. 6: Results of **GAT backbone** under **label-related distribution shift** for the task of semi-supervised node classification. Comparing with GAT method, our BA-GAT method (by applying our BA-GNN framework on GAT backbone) improves the accuracy of node classification across different label biased environments.

authors, which are connected by an edge if they co-authored a paper. Node features represent paper keywords for each author's papers. Each node has a label denoting the most active fields of study for the corresponding author.

Co-purchase datasets. Amazon Computers and Amazon Photo [51] are segments of the Amazon co-purchase graph [52] where nodes are goods and edges denote that two goods are frequently bought together. Node features are derived from bag-of-words representations for product reviews and class labels are given by the product category.

B. Baselines

Note that the proposed framework BA-GNN is generic and can be utilized to improve arbitrary GNN backbones. To evaluate the effectiveness of BA-GNN framework, we consider three popular GNN architectures, including GCN [1], GAT [2] and APPNP [7]. We implemented our proposed BA-GNN and some necessary baselines using Pytorch [53] and Pytorch Geometric [54], a library for deep learning on irregularly structured data built upon Pytorch.

Moreover, we further consider the methods that are designed for reducing the degree bias, including GNM [20], DEMO-Net [21], SL-DSGCN [22]. We aim to provide a rigorous and fair comparison between different models on each dataset by using the same dataset splits and training procedure. We tune hyperparameters for all models individually and some baselines even achieve better results than their original reports.

C. RQ1. Overall performance in Graph Benchmark

To show the effectivess of our proposed BA-GNN, we follow the widely used semi-supervised setting in [1]–[3], and apply the standard fixed training/validation/testing split as shown in Table I. We conduct node classification on 7 graph datasets as introduced in Table I with different backbones such as GCN, GAT and APPNP. For each graph dataset, we report the mean accuracy with standard deviaton on the test nodes

TABLE II: Classification accuracy (%) results of semi-supervised node classification experiments. We follow the widely used semi-supervised setting in [1]–[3], and apply the standard fixed training/validation/testing split as shown in Table I.

Method	Cora	CiteSeer	PubMed	Coauthor-CS	Coauthor-Physics	Amazon-Computers	Amazon-Photo
GCN	81.3±0.8	71.1±0.7	78.8±0.6	91.08±0.6	93.03±0.8	81.17±1.8	90.25±1.6
GCN+**Ours**	**82.1±0.5**	**72.4±0.6**	**79.8±0.4**	**92.73±0.6**	**94.94±0.1**	**84.94±2.4**	**91.39±1.1**
GAT	83.0±0.7	72.5±0.7	79.0 ±0.4	90.28±0.7	91.92±0.9	80.38±1.5	90.41±1.8
GAT+**Ours**	**83.5±0.7**	**73.4±0.5**	**80.3 ±0.3**	**91.34±0.5**	**92.39±0.8**	**81.29±2.2**	**91.53±1.4**
APPNP	83.8±0.3	71.6±0.5	79.7±0.3	92.45±0.4	93.41±0.9	81.79±2.0	91.22±1.3
APPNP+**Ours**	**84.2±0.3**	**72.0±0.5**	**79.9±0.3**	**92.94±0.5**	**94.36±1.1**	**82.64±1.9**	**91.81±1.3**

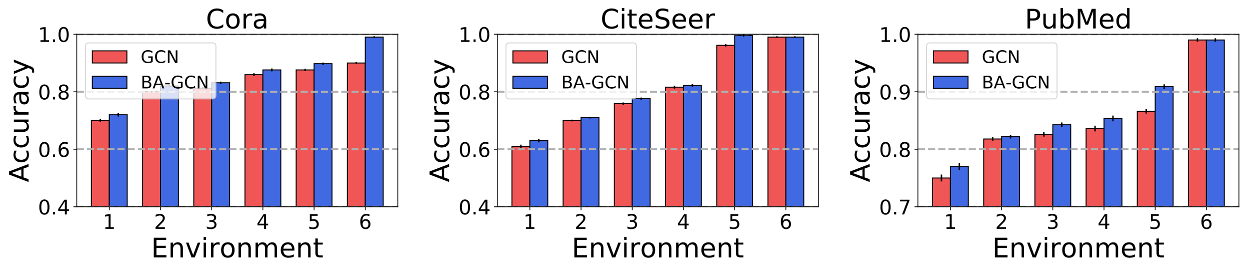


Fig. 7: Results of **GCN backbone** under **degree-related distribution shift** for the task of semi-supervised node classification. Comparing with GCN method, our BA-GCN method (by applying our BA-GNN framework on GCN backbone) improves the accuracy of node classification across different degree biased environments.

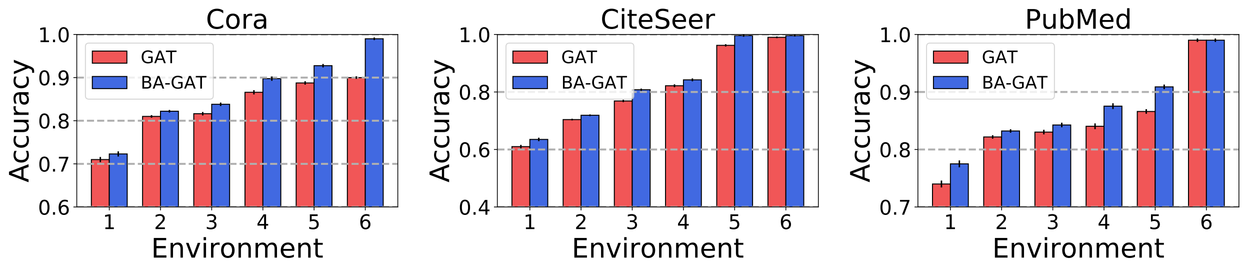


Fig. 8: Results of **GAT backbone** under **degree-related distribution shift** for the task of semi-supervised node classification. Comparing with GAT method, our BA-GAT method (by applying our BA-GNN framework on GAT backbone) improves the accuracy of node classification across different degree biased environments.

in Table II with 10 runs experiments. As shown in Table II, we have the following findings:

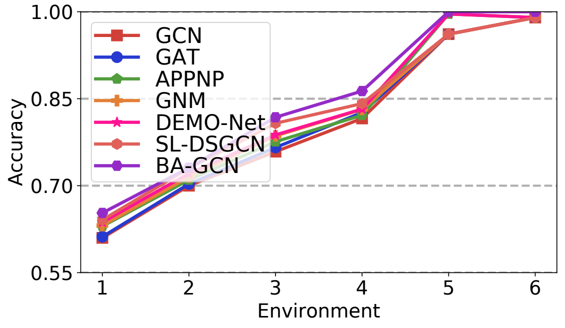
- Our BA-GNN improves the performance of all GNN backbones consistently and significantly in all settings.
- GNN backbones have worse performance due to ignoring of biases.
- The reason why our BA-GNNs outperform is that our method alleviates different biases, thus our method outperform in different bias environments, and the overall classification accuracy increases.
- In summary, our BA-GNN achieves superior performance on all these seven datasets, which significantly demonstrates the effectiveness of our proposed framework and our motivation.

D. RQ2. Performance in unknown bias environments.

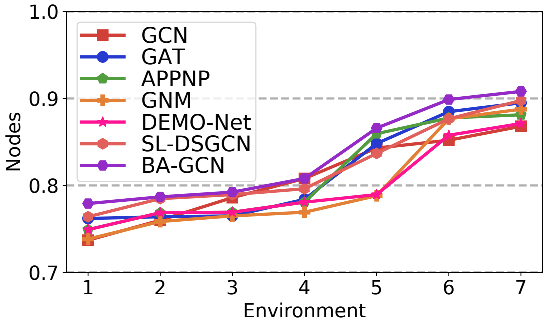
To further validate the effectiveness of our framework, besides the widely used semi-supervised setting in [1]–[3], we evaluate our framework in different biased unknown test environments. Specifically, for degree biased test environments, we group the testing nodes of each graph according to their degrees. Similarly, for label biased test environments, we group the testing nodes of each graph according to their labels. The distribution shifts between training and testing environments are various in this setting, where we evaluate methods in different environments rather than average accuracy of all testing nodes in a single environment. The training/validation/testing split in this experiment is the standard fixed split as shown in Table I, which is widely used in [1]–[3].

Compare with base backbone. The testing results in unknown environments are shown in Figure 5 - 8. Specifically, each figure in Figure 5 - 8 plot the evaluation results across different testing environments. From the results, we have the following observations and conclusions:

- Our BA-GNN outperforms all backbones such as GCN, GAT, APPNP in all bias cases.



(a) Unknown degree bias environments.



(b) Unknown label bias environments.

Fig. 9: Semi-supervised node classification accuracy compared with other GNN baselines for addressing the bias issue. We test all methods on different label-bias and degree-bias environments.

- GNN backbones have worse performance due to ignoring of biases.
- Results illustrate the effectiveness of our BA-GNN. The reason why BA-GNN outperform in all cases is that our method could alleviate different unknown biases with invariant representation.
- In summary, our BA-GNN outperforms the baselines with different backbones and different graphs, which suggests that our framework is agnostic to backbones and graphs.

Compare with previous methods designed for reducing biases. To further show the effectiveness of our framework, we compare our BA-GNN with the following baselines designed for reducing biases: GNM [20], DEMO-Net [21], SL-DSGCN [22]. Figure 9 shows the performance of different methods designed for reducing biases in unknown test environments. Specifically, Figure 9 (a) plots the evaluation results across different unknown testing environments with degree bias, and Figure 9 (b) plots the evaluation results across different unknown testing environments with label bias. From these figures, we have the following observations and findings:

- Our BA-GNN outperforms all methods in all bias cases, including methods are designed for reducing degree bias.
- Although the degree-related method such as DEMO-Net and SL-DSGCN can also alleviate the degree bias, they still do not outperform APPNP on label bias environments.
- Results illustrate the effectiveness of our BA-GNN. The reason why BA-GNN outperform in all cases is that our method could alleviate different biases, methods such as GNM, DEMO-Net and SL-DSGCN only focus on degree bias.
- Our BA-GCN is the only method that performs better than all baselines across all the environments. These findings show that our BA-GCN framework can effectively adapt to different graph and alleviate different bias.
- In summary, our BA-GNN could alleviate different biases such as degree bias and label bias. Thus our method outperforms the methods designed for reducing biases.

TABLE III: Training time (s) of different methods on the same GPU device.

Method	Cora	CiteSeer	PubMed
GCN	2.6	3.0	5.3
GAT	4.7	5.2	8.1
APPNP	3.1	3.7	6.8
GNM	4.9	5.7	9.4
DEMO-Net	5.6	6.0	8.7
SL-DSGCN	5.7	6.2	8.9
Ours	4.8	5.4	7.8

E. RQ.3 Effectiveness of Bias-Aware Environment Clustering

To evaluate if our BA-GNN can learn good bias identification, we study the bias identification for individual nodes. Figures 10 and 11 show the case studies of bias identification on both label bias and degree bias. In Figures 10 and 11, we plot the 3-hop neighborhood of each test node and use different colors to indicate different labels. The node with higher degree is assigned to the environment where degree of nodes is higher, as shown in Figure 10 (a), and the node with lower degree is assigned to the environment where degree of nodes is lower, as shown in Figure 10 (b). Similarly, As shown in Figure 11, we find that the node with more same class neighbors tends to be assigned to environment where label has more nodes. In contrast, the node with fewer same class nodes will probably be assigned to environment where label has fewer nodes.

Additionally, we can find that our framework successfully identifies the bias.

F. RQ.4 Efficiency of BA-GNN

Training time analysis. To further show the efficiency of our BA-GNN, we compare our approach with baselines on the same device as shown in Table III. The computational costs is only slightly increased compared with base backbone. Our BA-GNN is more efficiency than previous methods that aim at reduce degree bias.

VII. Conclusion

In this paper, we argue that GNNs might suffer from the bias issue because of the distribution shift between training

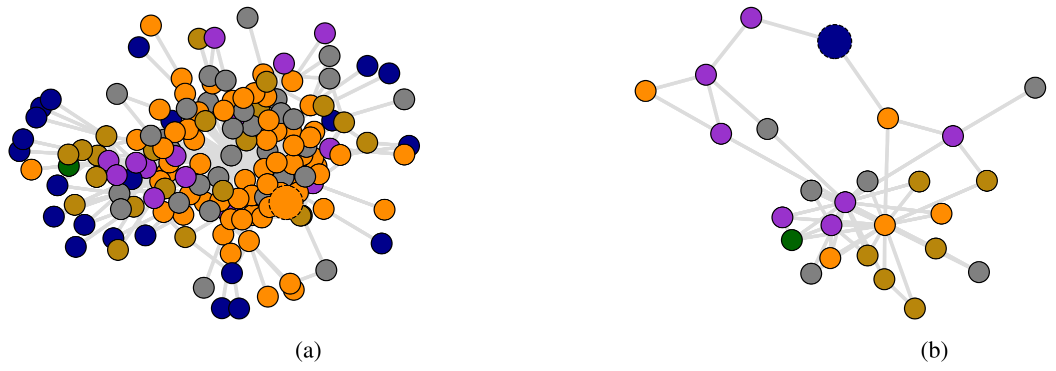


Fig. 10: Illustration of the inferred environments for the degree bias by our BA-GNN. The node assigned to environment is the bigger node in each sub-graph.(a) Node in Environment 6, where most of nodes are high-degree. (b) Node in Environment 1, where most of nodes are low-degree.

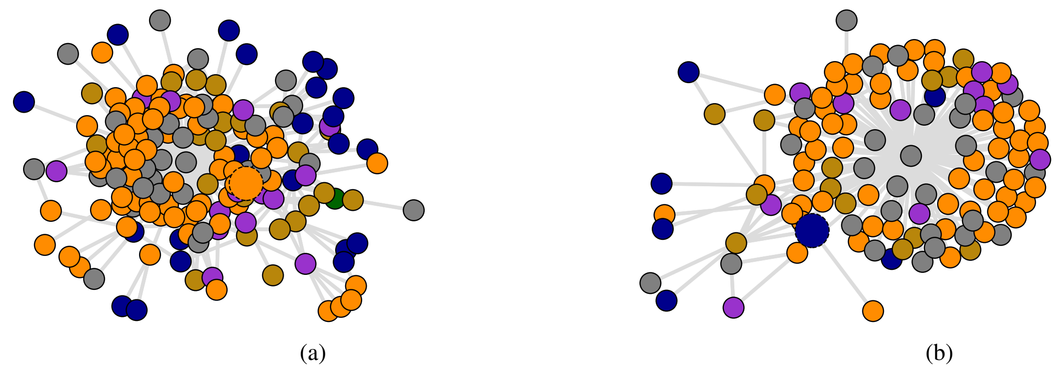


Fig. 11: Illustration of the inferred environments for the label bias by our BA-GNN. The node assigned to environment is the bigger node in each sub-graph. (a) Node in Environment 6, where node's label has more nodes. (b) Node in Environment 1, where node's label has fewer nodes.

and testing node distributions. More importantly, the test node distribution in the graph is generally unknown during mode training in real-world applications. To address this problem, we propose a novel *Bias-Aware Graph Neural Network* (BA-GNN) framework that aims to learn node representations that are invariant across different distributions for invariant prediction. Our BA-GNN framework contains two interactive parts, one for bias identification and the other for invariant prediction. To learn invariant feature and aggregated representation, our BA-GNN learns multiple biased graph partitions and selects feature, neighbor, and propagation steps for nodes under multiple biased graph partitions. With extensive empirical experiments on different graphs and different GNN backbones, we demonstrate the effectiveness of our proposed framework, which leads to the state-of-the-art performance on several benchmark datasets.

References

[1] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th ICLR*, 2017.

[2] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *6th ICLR*, 2018.

[3] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *NIPS*, 2016.

[4] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *31st NeurIPS*, 2017, pp. 1025–1035.

[5] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.

[6] F. Errica, M. Podda, D. Bacciu, and A. Micheli, "A fair comparison of graph neural networks for graph classification," in *ICLR*, 2020.

[7] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," in *7th ICLR 2019*, 2019.

[8] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, "Simple and deep graph convolutional networks," in *ICML*. PMLR, 2020, pp. 1725–1735.

[9] 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.

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

[11] J. Wu, X. Wang, F. Feng, X. He, L. Chen, J. Lian, and X. Xie, "Self-supervised graph learning for recommendation," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 726–735.

[12] J. You, B. Liu, R. Ying, V. Pande, and J. Leskovec, "Graph convolutional policy network for goal-directed molecular graph generation," *arXiv preprint arXiv:1806.02473*, 2018.

[13] D. Wang, J. Lin, P. Cui, Q. Jia, Z. Wang, Y. Fang, Q. Yu, J. Zhou, S. Yang, and Y. Qi, "A semi-supervised graph attentive network for financial fraud detection," in *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2019, pp. 598–607.

[14] X. Wang, Y. Ma, Y. Wang, W. Jin, X. Wang, J. Tang, C. Jia, and J. Yu,

"Traffic flow prediction via spatial temporal graph neural network," in *Proceedings of The Web Conference 2020*, 2020, pp. 1082–1092.

[15] C. Huang, X. Wu, X. Zhang, C. Zhang, J. Zhao, D. Yin, and N. V. Chawla, "Online purchase prediction via multi-scale modeling of behavior dynamics," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 2613–2622.

[16] Y. He, P. Cui, J. Ma, H. Zou, X. Wang, H. Yang, and P. S. Yu, "Learning stable graphs from multiple environments with selection bias," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 2194–2202.

[17] 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*, 2020.

[18] M. Meila and T. Zhang, Eds., *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, ser. Proceedings of Machine Learning Research, vol. 139. PMLR, 2021. [Online]. Available: http://proceedings.mlr.press/v139/

[19] 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.

[20] F. Zhou, T. Li, H. Zhou, J. Ye, and H. Zhu, "Graph-based semi-supervised learning with nonignorable nonresponses," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 2019, pp. 7015–7025.

[21] J. Wu, J. He, and J. Xu, "Net: Degree-specific graph neural networks for node and graph classification," in *25th ACM SIGKDD KDD*, 2019, pp. 406–415.

[22] X. Tang, H. Yao, Y. Sun, Y. Wang, J. Tang, C. Aggarwal, P. Mitra, and S. Wang, "Investigating and mitigating degree-related biases in graph convoltuional networks," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 1435–1444.

[23] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *International conference on machine learning*. PMLR, 2015, pp. 1180–1189.

[24] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7167–7176.

[25] S. Sankaranarayanan, Y. Balaji, C. D. Castillo, and R. Chellappa, "Generate to adapt: Aligning domains using generative adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8503–8512.

[26] P. M. Esfahani and D. Kuhn, "Data-driven distributionally robust optimization using the wasserstein metric: performance guarantees and tractable reformulations," *Math. Program.*, vol. 171, no. 1-2, pp. 115–166, 2018. [Online]. Available: https://doi.org/10.1007/s10107-017-1172-1

[27] J. Duchi and H. Namkoong, "Learning models with uniform performance via distributionally robust optimization," *arXiv preprint arXiv:1810.08750*, 2018.

[28] A. Sinha, H. Namkoong, and J. Duchi, "Certifying some distributional robustness with principled adversarial training," *International Conference on Learning Representations*, 2018.

[29] S. Sagawa, P. W. Koh, T. B. Hashimoto, and P. Liang, "Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization," *arXiv preprint arXiv:1911.08731*, 2019.

[30] W. Hu, G. Niu, I. Sato, and M. Sugiyama, "Does distributionally robust supervised learning give robust classifiers?" in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, ser. Proceedings of Machine Learning Research, J. G. Dy and A. Krause, Eds., vol. 80. PMLR, 2018, pp. 2034–2042. [Online]. Available: http://proceedings.mlr.press/v80/hu18a.html

[31] C. Frogner, S. Claici, E. Chien, and J. Solomon, "Incorporating unlabeled data into distributionally robust learning," *arXiv preprint arXiv:1912.07729*, 2019.

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

[33] M. Koyama and S. Yamaguchi, "Out-of-distribution generalization with maximal invariant predictor," *CoRR*, vol. abs/2008.01883, 2020. [Online]. Available: https://arxiv.org/abs/2008.01883

[34] S. Chang, Y. Zhang, M. Yu, and T. S. Jaakkola, "Invariant rationalization," in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, ser. Proceedings of Machine Learning Research, vol. 119. PMLR, 2020, pp. 1448–1458. [Online]. Available: http://proceedings.mlr.press/v119/chang20c.html

[35] E. Creager, J.-H. Jacobsen, and R. Zemel, "Environment inference for invariant learning," in *ICML Workshop on Uncertainty and Robustness*, 2020.

[36] J. Liu, Z. Hu, P. Cui, B. Li, and Z. Shen, "Heterogeneous risk minimization," in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, 2021, pp. 6804–6814. [Online]. Available: http://proceedings.mlr.press/v139/liu21h.html

[37] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and deep locally connected networks on graphs," in *2nd ICLR 2014*, 2014.

[38] H. Pei, B. Wei, K. C.-C. Chang, Y. Lei, and B. Yang, "Geom-gcn: Geometric graph convolutional networks," in *ICLR*, 2019.

[39] X. Tang, H. Yao, Y. Sun, Y. Wang, J. Tang, C. C. Aggarwal, P. Mitra, and S. Wang, "Investigating and mitigating degree-related biases in graph convoltuional networks," in *The 29th ACM International Conference on Information CIKM*, 2020, pp. 1435–1444.

[40] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *ICLR*, 2018.

[41] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Advances in neural information processing systems*, 2016.

[42] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv:1710.10903*, 2017.

[43] J. Zhang, X. Shi, J. Xie, H. Ma, I. King, and D.-Y. Yeung, "Gaan: Gated attention networks for learning on large and spatiotemporal graphs," *arXiv:1803.07294*, 2018.

[44] M. Long, Z. Cao, J. Wang, and M. I. Jordan, "Conditional adversarial domain adaptation," *arXiv preprint arXiv:1705.10667*, 2017.

[45] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. Efros, and T. Darrell, "Cycada: Cycle-consistent adversarial domain adaptation," in *International conference on machine learning*. PMLR, 2018, pp. 1989–1998.

[46] K. Saito, K. Watanabe, Y. Ushiku, and T. Harada, "Maximum classifier discrepancy for unsupervised domain adaptation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3723–3732.

[47] 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.

[48] A. El Gamal and Y.-H. Kim, "Network information theory," *Network Information Theory*, 12 2011.

[49] Y. Yamada, O. Lindenbaum, S. Negahban, and Y. Kluger, "Feature selection using stochastic gates," in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, ser. Proceedings of Machine Learning Research, vol. 119. PMLR, 2020, pp. 10 648–10 659. [Online]. Available: http://proceedings.mlr.press/v119/yamada20a.html

[50] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, pp. 93–93, 2008.

[51] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, "Pitfalls of graph neural network evaluation," *arXiv preprint arXiv:1811.05868*, 2018.

[52] J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel, "Image-based recommendations on styles and substitutes," in *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, 2015, pp. 43–52.

[53] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[54] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," *arXiv preprint arXiv:1903.02428*, 2019.