Debiased Graph Neural Networks with Agnostic Label Selection Bias

Shaohua Fan, Xiao Wang, *Member, IEEE,* Chuan Shi†, *Member, IEEE,* Kun Kuang, Nian Liu, Bai Wang

***Abstract*—Most existing Graph Neural Networks (GNNs) are proposed without considering the selection bias in data, i.e., the inconsistent distribution between the training set with test set. In reality, the test data is not even available during the training process, making selection bias agnostic. Training GNNs with biased selected nodes leads to significant parameter estimation bias and greatly impacts the generalization ability on test nodes. In this paper, we first present an experimental investigation, which clearly shows that the selection bias drastically hinders the generalization ability of GNNs, and theoretically prove that the selection bias will cause the biased estimation on GNN parameters. Then to remove the bias in GNN estimation, we propose a novel Debiased Graph Neural Networks (DGNN) with a differentiated decorrelation regularizer. The differentiated decorrelation regularizer estimates a sample weight for each labeled node such that the spurious correlation of learned embeddings could be eliminated. We analyze the regularizer in causal view and it motivates us to differentiate the weights of the variables based on their contribution on the confounding bias. Then, these sample weights are used for reweighting GNNs to eliminate the estimation bias, thus help to improve the stability of prediction on unknown test nodes. Comprehensive experiments are conducted on several challenging graph datasets with two kinds of label selection biases. The results well verify that our proposed model outperforms the state-of-the-art methods and DGNN is a flexible framework to enhance existing GNNs.**

***Index Terms*—Graph Neural Networks, Casual Inference, Selection Bias.**

I. Introduction

Graph Neural Networks (GNNs) are powerful deep learning algorithms on graphs with various applications [1], [2], [3], [4]. Existing GNNs mainly learn a node embedding through aggregating the features from its neighbors, and such message-passing framework is supervised by the node label in an end-to-end manner. During this training procedure, GNNs will effectively learn the correlation between the structure patterns and node features with the node labels, so that GNNs are capable of learning the embeddings of new nodes and inferring their labels.

One basic requirement of GNNs making precise predictions on unseen test nodes is that the distribution of labeled training nodes and test nodes is the same, i.e., the structure and feature of labeled training and test nodes follow the similar pattern, so that the learned correlation between the current graph and labels can be well generalized to the new nodes. However, in reality, there are two inevitable issues. (1) Because it is difficult to control the graph collection in an unbiased manner, the relationship between the collected real-world graph and the labeled nodes is inevitably biased. Training on such a graph will cause biased correlation with node labels. Taking a scientist collaboration network as an example, if most scientists with "machine learning" (ML) label collaborate with those with "computer vision" (CV) label, existing GNNs may learn spurious correlation, i.e., scientists who cooperate with CV scientists are ML scientists. If a new ML scientist only connects with ML scientists or the scientists in other areas, it will be probably misclassified. (2) The test nodes in the real scenario are usually not available in the training phase, implying that the distribution of new nodes is agnostic. Once the distribution is inconsistent with that in the training nodes, the performance of all the current GNNs will be hindered. Even transfer learning is able to solve the distribution shift problem, however, it still needs the prior of test distribution, which actually cannot be obtained beforehand. Therefore, the agnostic label selection bias greatly affects the generalization ability of GNNs on unknown test data.

In order to observe selection bias in real graph data, we conduct an experimental investigation to validate the effect of selection bias on GNNs (see Section II-A). We select training nodes with different biased degrees for each dataset, making the distribution of training nodes and test nodes inconsistent. The results clearly show that selection bias drastically hinders the performance of GNNs on unseen test nodes. Moreover, with heavier bias, the performance drops more. Further, we theoretically analyze how the data selection bias results in the estimation bias in GNN parameters (see Section II-B). Based on the stable learning technique [5], we can assume that the learned embeddings consist of two parts: stable variables and unstable variables. The data selection bias will cause spurious correlation between these two kinds of variables. Thereby we prove that with the inevitable model misspecification, the spurious correlation will further cause the parameter estimation bias. Once the weakness of the current GNNs with selection bias is identified, one natural question is "*how to remove the estimation bias in GNNs?*"

In this paper, we propose a novel Debiased Graph Neural Network (DGNN) framework for stable graph learning by jointly optimizing a differentiated decorrelation regularizer and a weighted GNN model. Specifically, the differentiated

† Corresponding author.

S. Fan, N. Liu and B. Wang are with the Department of Computer Science, Beijing University of Posts and Telecommunications, Beijing, China. E-mail: {fanshaohua, nianliu, wangbai}@bupt.edu.cn

X. Wang and C. Shi are with the Key Laboratory of Trustworthy Distributed Computing and Service (MoE), Beijing University of Posts and Telecommunications, Beijing, China and the Peng Cheng Laboratory, Shenzhen, China. E-mail: {xiaowang,shichuan}@bupt.edu.cn

K. Kuang is with the College of Computer Science and Technology, Zhejiang University, Zhejiang, China. E-mail: kunkuang@zju.edu.cn

Manuscript received 05 Feb. 2021; revised 02 Sep. 2021; accepted 29 Dec 2021.

decorrelation regularizer is able to learn a set of sample weights under differentiated variable weights, so that the spurious correlation between stable and unstable variables would be greatly eliminated. Based on the causal view analysis of the decorrelation regularizer, we theoretically prove that the weights of variables can be differentiated by the regression coefficients. Compared with existing decorrelation methods [5], [6], the proposed regularizer is able to remove the spurious correlation while maintaining a higher effective sample size and requiring less prior knowledge. Moreover, to better combine the decorrelation regularizer with existing GNN architecture, the theoretical result shows that adding the regularizer to the embeddings learned by the penultimate layer could be both theoretically sound and flexible. Then the sample weights learned by the decorrelation regularizer are used to reweight the GNN loss so that the parameter estimation could be unbiased.

In summary, the contributions of this paper are three-fold: i) We investigate a new problem of learning GNNs with agnostic label selection bias. The problem setting is general and practical for real applications. ii) We bring the idea of variable decorrelation into GNNs to relieve bias influence on model learning and propose a general framework DGNN that could be adopted to various GNNs. iii) We conduct experiments on real-world graph benchmarks with two kinds of agnostic label selection biases, and the experimental results demonstrate the effectiveness and flexibility of our model.

II. Effect of Label Selection Bias on GNNs

In this section, we first summarize the main notations used in this paper in Table I and then formulate our target problem:

Problem 1 (**Semi-supervised Learning on Graph with Agnostic Label Selection Bias**). *Given a training graph* $G_{train} = (\mathbf{A}_{train}, \mathbf{X}_{train}, \mathbf{Y}_{train})$, *where* $\mathbf{A}_{train} \in \mathbb{R}^{N \times N}$ *(*N *nodes) represents the adjacency matrix,* $\mathbf{X}_{train} \in \mathbb{R}^{N \times D}$ *(*D *features) refers to the node feature vectors and* $\mathbf{Y}_{train} \in \mathbb{R}^{n \times C}$ *(*n *labeled nodes,* C *classes) refers to the available labels for training (*$n \leq N$*), the task is to learn a GNN* g *with parameter* θ *to precisely predict the label of nodes on test graph* $G_{test} = (\mathbf{A}_{test}, \mathbf{X}_{test}, \mathbf{Y}_{test})$, *where distribution* $G_{train} \neq G_{test}$.

A. Experimental Investigation

We conduct an experimental investigation to examine whether the existing GNNs are sensitive to the distribution shifts caused by the label selection bias. One motivating example is that due to the research interests of the researcher that they are more likely to label the interdisciplinary documents that cite more papers from different subjects in a citation network, while testing may be conducted on nodes with any neighborhood distribution. Based on this example, the main idea of the experimental investigation is that we will perform two representative GNNs: GCN [2] and GAT [3] on three widely used graph datasets: *Cora*, *Citeseer*, *Pubmed* [7] with different bias degrees. If the performance drops sharply comparing with the scenarios without selection bias, this will demonstrate that GNNs cannot generalize well in selection bias settings.

TABLE I: Glossary of Notations.

Notation	Description
$\mathcal{G}_{train}$	Training graph
$\mathcal{G}_{test}$	Test graph
$\mathbf{A}_{train/test}$	The adjacency matrix of $\mathcal{G}_{train}$ or $\mathcal{G}_{test}$
$\mathbf{X}_{train/test}$	The node feature vectors of $\mathcal{G}_{train}$ or $\mathcal{G}_{test}$
$\mathbf{Y}_{train/test}$	The node label vectors of $\mathcal{G}_{train}$ or $\mathcal{G}_{test}$
$\mathbf{H}/\hat{G}(\mathbf{X}, \mathbf{A}, g)$	Node embeddings matrix learned by GNNs
$\mathbf{S}$	The stable variables in $\mathbf{H}$
$\mathbf{V}$	The unstable variables in $\mathbf{H}$
$\mathbf{S}$	The latent stable variables to generate $\mathbf{Y}$
$\mathbf{V}$	The unstable variables to generate label $\mathbf{Y}$
$\tilde{\gamma}_S$	The linear coefficients for $\mathbf{S}$
$\tilde{\gamma}_V$	The linear coefficients for $\mathbf{V}$
g	The non-linear transformation for stable variables $\mathbf{S}$
γ_S	The linear coefficients for $\mathbf{S}$
γ_V	The linear coefficients for $\mathbf{V}$
T	Treatment variable
$\mathbf{X}$	Confounders
$\mathbf{w}$	Sample weights
	Variable weights in DVD term
	The linear coefficients for confounders $\mathbf{X}$
	The linear coefficient for treatment T
$Y_i(t)$	The potential outcome of sample i with treatment t

To simulate the agnostic selection bias scenario, we first follow the inductive setting in [8] that masks the validation and test nodes as the training graph G_{train} in the training phase, and then infer the labels of validation and test nodes with whole graph G_{test}. In this way, the distribution of test node can be considered agnostic. Following [9], we design a biased label selection method on training graph G_{train}. The selection variable e is introduced to control whether the node will be selected as labeled nodes, where $e = 1$ means selected and 0 otherwise. For node i, we calculate its neighbor distribution ratio: $r_i = |\{j : j \in N_i, y_j \neq y_i\}| / |N_i|$, where N_i is neighborhood of node i in G_{train} and $y_j \neq y_i$ means the label of central node i is not the same as the label of its neighborhood node j. And r_i measures the difference between the label of central node i with the labels of its neighborhood. Then we average all the nodes' r to get a threshold t. For each node, the probability to be selected is: $P(e_i = 1 \mid r_i) = \begin{cases} \rho & r_i < t \\ 1 & r_i \geq t \end{cases}$, where $\rho \in [0.5, 1]$ is used to control the degree of selection bias and a larger ρ means a heavier bias. We set ρ as {0.7, 0.8, 0.9} to get three bias degrees for each dataset, termed as *Light, Medium, Heavy*, respectively. We select 20 nodes for each class for training and the validation and test nodes are the same as [10]. Furthermore, we take the *unbiased* datasets as baselines, where the labeled nodes are selected randomly.

Figure 1 is the results of GCN, GAT and our proposed method, GCN/GAT-DVD, on these datasets with four bias degrees. We can find that: i) Compared with the unbiased scenario, when performing GCN/GAT on biased datasets, they suffer from serious performance decrease, indicating that selection bias greatly affects the GNNs' performance. ii) All lines decrease monotonically with the increase of bias degree, demonstrating that heavier biases will cause larger performance reduction. iii) GCN/GAT-DVD outperforms the corresponding base models (i.e., GCN/GAT) consistently and achieves larger improvements in heavier bias degree scenarios, indicating that our proposed method could relieve the effect of selection bias.

(a) Cora (b) Citeseer (c) Pubmed

Fig. 1: Effect of selection bias on GCN and GAT.

B. Theoretical Analysis

The above experiment empirically verifies the effect of selection bias on GNNs. Here we theoretically analyze the effect of selection bias on estimating the parameters in GNNs. First, because biased labeled nodes have biased neighborhood structure and features, GNNs will encode this biased information into the node embeddings, which is validated by the experimental investigation. Based on stable learning technique [5], we make the following assumption:

Assumption 1. The node embeddings learned by GNNs for each node can be decomposed as $r(S, V)$, where S represents the stable variables and V represents the unstable variables. Specifically, for both training and test environments, $E(Y|S=s, V=v) = E(Y|S=s)$.

Under Assumption 1, the distribution shift between training set and test set is mainly induced by the variation in the joint distribution over (S, V), i.e., $P(S_{train}, V_{train}) \neq P(S_{test}, V_{test})$. However, there is an invariant relationship between stable variables S and outcome Y in both training and test environments, which can be expressed as $P(Y_{train}|S_{train}) = P(Y_{test}|S_{test})$. Assumption 1 can be guaranteed by $Y \perp V|S$. Thus, one can solve the stable prediction problem by developing a function g based on S. However, one can hardly identify such variables in GNNs.

Without loss of generality, we take Y as a continuous variable for analysis and have the following assumption:

Assumption 2. The true generation process of target variable Y contains not only the linear combination of stable variables S, but also the nonlinear transformation of stable variables.

Based on the above assumptions, we formalize the label generation process as follows:

$$Y = f(X, A) = S_S + V_V + g(S) + \varepsilon, \quad (1)$$

where S and V are latent stable variables and unstable variables to generate label Y, which can be learned by GNNs from raw graph data, $_S$ and $_V$ are the corresponding linear coefficients and they represent the effect of each latent variable on outcome Y, ε is the independent random noise, and g is the nonlinear transformation function of stable variables. According to Assumption 1, we know that coefficients of unstable variables V are actually 0 (i.e., $_V = 0$).

For a classical GNN model with a linear regression predictor, its prediction function can be formulated as:

$$\hat{Y} = \hat{G}(X, A; \hat{}_g)\hat{}_S + \hat{G}(X, A; \hat{}_g)\hat{}_V + \varepsilon, \quad (2)$$

where $\hat{G}(X, A; \hat{}_g) \in \mathbb{R}^{N \times p}$ denotes the node embeddings learned by a GNN, such as GCN and GAT, the output variables of $\hat{G}(X, A; \hat{}_g)$ can be decomposed as stable variables $\hat{G}(X, A; \hat{}_g)_S \in \mathbb{R}^{N \times m}$ and unstable variables $\hat{G}(X, A; \hat{}_g)_V \in \mathbb{R}^{N \times q}$ $(m + q = p)$ corresponding to S and V in Eq. (1). Compared with Eq. (1), we can find that the parameters of GNN model could be unbiasedly estimated if the nonlinear term $g(S) = 0$ (i.e., there does not exist any nonlinear relationship in the label generation process that cannot be learned by GNNs), because the GNN model in Eq. (2) will have the same label generation mechanism as Eq. (1). However, as common used GNNs only have several layers which may limit their nonlinear power and the real-world graph data is far more complicated, it is reasonable to assume that there is a nonlinear term $g(S) \neq 0$ that cannot be fitted by the GNNs. Under this assumption, next, we taking a vanilla GCN [2] as an example to illustrate how the distribution shift will induce parameter estimation bias. A two-layer GCN can be formulated as $\hat{A}\,\sigma(\hat{A}XW^{(0)})W^{(1)}$, where $\hat{A}$ is the normalized adjacency matrix, W is the transformation matrix at each layer and σ is the Relu activation function. We decompose GCN as two parts: one is embedding learning part $\hat{A}\,\sigma(\hat{A}XW^{(0)})$, which can be decomposed as $(S^T, V^T)^T$, corresponding to $\hat{G}(X, A; \hat{}_g)_S$ and $\hat{G}(X, A; \hat{}_g)_V$ in Eq. (2), and the other part is $W^{(1)}$, where the learned parameters can be decomposed as $\tilde{}_S, \tilde{}_V$, corresponding to $\hat{}_S, \hat{}_V$ in Eq. (2). We aim at minimizing the least-square loss:

$$L_{GCN} = \sum_{i=1}^{n} (S_i^T \tilde{}_S + V_i^T \tilde{}_V - Y_i)^2. \quad (3)$$

According to the derivation rule of partitioned regression model [5], [11], with $S = S$ and $V = V$, we have:

$$\begin{aligned} \tilde{}_V = {} & \left(\frac{1}{n}\sum_{i=1}^{n} V_i^T V_i\right)^{-1} \frac{1}{n}\sum_{i=1}^{n} V_i^T g(S_i) \\ & \left(\frac{1}{n}\sum_{i=1}^{n} V_i^T V_i\right)^{-1} \frac{1}{n}\sum_{i=1}^{n} V_i^T S_i \quad _S - \tilde{}_S, \end{aligned} \quad (4)$$

$$\tilde{\gamma}_S \quad \gamma_S \quad \frac{1}{n}\sum_{i=1}^{n} S_i^T S_i \quad ^{1} \frac{1}{n}\sum_{i=1}^{n} S_i^T g\ S_i$$
$$\frac{1}{n}\sum_{i=1}^{n} S_i^T S_i \quad ^{1} \frac{1}{n}\sum_{i=1}^{n} S_i^T V_i \quad _V \quad \tilde{\gamma}_V \ ; \tag{5}$$

where n is labeled node size, S_i is i-th sample of S, $\frac{1}{n}\sum_{i=1}^{n} V_i^T g\ S_i \quad E\ V^T g\ S \quad o_p\ 1$, $\frac{1}{n}\sum_{i=1}^{n} V_i^T S_i \quad E\ V^T S \quad o_p\ 1$ and $o_p\ 1$ is the error which is negligible. Ideally, $\tilde{\gamma}_V \quad _V \quad 0$ indicates that there is no bias between the estimated and the real parameter. However, if $E\ V^T S \ j\ 0$ or $E\ V^T g\ S \ j\ 0$ in Eq. (4), $\tilde{\gamma}_V$ will be biased, leading to the biased estimation on $\tilde{\gamma}_S$ in Eq. (5) as well, i.e, the true effect of learned embeddings on label can not be estimated precisely. Since the correlation between V and S (or $g\ S$) [1] might shift in test phase, the biased parameters learned in training set is not the optimal parameters for predicting testing nodes. Therefore, to increase the stability of prediction, we need to unbiasedly estimate the parameters $\tilde{\gamma}_V$ of V by removing the correlation between V and S (or $g\ S$) on training graph, making $E\ V^T S \quad 0$ and $E\ V^T g\ S \quad 0$. Note that $\frac{1}{n}\sum_{i=1}^{n} S_i^T g\ S_i$ in Eq. (5) can also cause estimation bias, but the relation between S and $g\ S$ is stable across environments, which do not affect the stability to some extent.

III. Proposed Model

A. Revisiting on Variable Decorrelation in Causal View

To decorrelate V and S (or $g\ S$), we should decorrelate the output variables of $G\ X; A; \ _g$. [5] proposes a Variable Decorrelation (VD) term with sample reweighting technique to eliminate the correlation between each variable pair, in which the sample weights are learned by jointly minimizing the moment discrepancy between each variable pair:

$$L_{VD}\ H \quad = \sum_{j=1}^{p} \| H_{:j}^T w H_{:\ j} \| H_{:j}^T w \| H_{:\ j}^T w \|_2^2 ; \tag{6}$$

where $H \ " \ R^{n \ p}$ means the variables needed to be decorrelated, i.e., $G\ X; A; \ _g$ of GNNs, $H_{:j}$ is j-th variable of H, $H_{:\ j}$ $H \ H_{:j}$ means all the remaining variables by setting the value of j-th variable in H as zero, $w \ " \ R^{n \ 1}$ is the sample weights, $\sum_{i=1}^{n} w_i \quad n$ and $w \quad diag\ w_1; \ ; w_n$ is the corresponding diagonal matrix. As we can see, $L_{VD}\ H$ can be reformulated as $\sum_{j \ k} \| H_{:j}^T w H_{:k} \| H_{:j}^T w \| H_{:k}^T w \|_2^2$, and it aims to let $E\ H_{:j}^T H_{:k} \quad E\ H_{:j}^T\ E\ H_{:k}$ for each variable pair j and k. $L_{VD}\ H$ decorrelates all the variable pairs equally. However, decorrelating all the variables requires sufficient samples [3], i.e., $n \quad \infty$, which is hard to be satisfied, especially in the semi-supervised setting. In this scenario, we cannot guarantee $L_{VD}\ H \quad 0$. Therefore, the key challenge is the difficulties of removing the spurious correlation that has the largest impact on the unbiased estimation when $L_{VD}\ H \ j\ 0$.

Inspired by confounding balancing technique in observational studies [12], we revisit the VD regularizer in causal view and show how to differentiate each variable pair. Confounding balancing techniques are often used for causal effect estimation of treatment T, where the distributions of confounders X are different between treated ($T \quad 1$) and control ($T \quad 0$) groups because of non-random treatment assignment. One could balance the distribution of confounders between treatment and control groups to unbiasedly estimate causal treatment effect [13], [14], [15]. Most balancing approaches exploit moments to characterize distributions, and balance them by adjusting sample weights w as follows: $w \quad \arg\min_w \| \sum_{i \ T_i \quad 1} X_i \ \sum_{i \ T_i \quad 0} w_i X_i \|_2^2$. After balancing, the treatment T and the confounders X tend to be independent.

Given one targeted variable $H_{:j}$, its decorrelation term $L_{VD_j} \quad \| H_{:j}^T w H_{:\ j} \| H_{:j}^T w \| H_{:\ j}^T w \|_2^2$, is to make $H_{:j}$ independent from $H_{:\ j}$ [2], which is same as the confounding balancing term making treatment and confounders independent. Thereby, L_{VD_j} can also be viewed as a confounding balancing term, where $H_{:j}$ is treatment and $H_{:\ j}$ is confounders, illustrated in Fig. 2(a). Hence, our target can be explained as unbiased estimation of causal effect of each variable which is invariant across training and test set. As different variable may contribute unequally to the confounding bias, it is necessary to differentiate the confounders. The target of differentiating confounders exactly matches our target that removing the correlation of variables that has the largest impact on the unbiased estimation.

B. Differetiated Variable Decorrelation

Considering a continuous treatment, the causal effect of treatment can be measured by Marginal Treatment Effect Function (MTEF) [16], and defined as: $MTEF \quad \frac{E\ Y_i\ t \quad E\ Y_i\ t \quad t}{t}$, where $Y_i\ t$ represents the potential outcome of sample i with treatment status $T \quad t$, and t denotes the increasing level of treatment. With the sample weights w decorrelating the treatment and the confounders, we can estimate the MTEF by:

$$MTEF \quad \frac{\frac{1}{n_i}\sum_{i \ T_i \quad t} w_i \quad Y_i \quad T_i \quad \frac{1}{n_j}\sum_{j \ T_j \quad t \quad t} w_j \quad Y_j \quad T_j}{t} ; \tag{7}$$

where n_i and n_j are the number of samples for two groups, respectively. Next, we theoretically analyze how to differentiate confounders' weights with the following theorem.

Theorem 1. *In observational studies, different confounders make unequal confounding bias on Marginal Treatment Effect Function (MTEF) with their own weights, and the weights can be learned via regressing outcome Y on confounders X and treatment variable T.*

Proof. Recalling the Assumption 1, we rewrite the label generation process Eq. (1) under MTEF setting as:

$$Y\ t \quad = \sum_k \ _k X_{:k} \quad t \quad g_t\ S \quad " ; \tag{8}$$

[1] We assume all variables are centered with zero mean. This assumption could be satisfied by adding a normalization layer after the learned embeddings.

[2] Nonlinear relationships between variables can be incorporated by considering high-order moments in Eq. (6), for example, a polynomial augmented function $H \quad H; H^2; H_{:i} H_{:j}; H^3; H_{:i} H_{:j} H_{:k}; \ .$

Fig. 2: (a) Diagram of decorrelating node embeddings with confounding balance. H^{K-1} is the node embedding matrix to be decorrelated. T is the treatment variable, corresponding to one target variable in H^{K-1}. X means the confounders, corresponding to the remaining variables of the target variable in H^{K-1}. Y is the outcome, corresponding to labels. (b) The framework of GNN-DVD. The same color in the two figures represents the same kind of variable.

where α ; β are the linear coefficients, and $g_t(S)$ is the output of nonlinear transformation of the stable variables S when treatment T is t. Note that if $T \perp S$, changing the value of T will not change the value of $g_t(S)$, $T \not\perp S$ otherwise.

Under above formulation, we write estimator of MTEF as:

$$
\begin{aligned}
\widehat{\mathrm{MTEF}} &= \frac{\frac{1}{n_i}\sum_{i:T_i=t} w_i Y_i - T_i \quad \frac{1}{n_j}\sum_{j:T_j=t} w_j Y_j - T_j}{t} \\
&\frac{\frac{1}{n_i}\sum_{i:T_i=t} w_i \sum_k \beta_k X_{ik} + g_t(S_i)}{t} \\
&\frac{\frac{1}{n_j}\sum_{j:T_j=t} w_j \sum_k \beta_k X_{jk} + g_t(S_i)}{t} \\
&\frac{\frac{1}{n_i}\sum_{i:T_i=t} w_i + \frac{1}{n_j}\sum_{j:T_j=t} w_j}{t} \\
&\frac{\frac{1}{n_i}\sum_{i:T_i=t} w_i \sum_k \beta_k X_{ik} \quad \frac{1}{n_j}\sum_{j:T_j=t} w_j \sum_k \beta_k X_{ik}}{t} \\
&\frac{\frac{1}{n_i}\sum_{i:T_i=t} w_i g_t(S_i) \quad \frac{1}{n_j}\sum_{j:T_j=t} w_j g_t(S_i)}{t} \\
\widehat{\mathrm{MTEF}} &= \sum_k \beta_k \frac{\sum_{i:T_i=t} \frac{1}{n_i} w_i X_{ik} \quad \sum_{j:T_j=t} \frac{1}{n_j} w_j X_{jk}}{t} \\
&\frac{\frac{1}{n_i}\sum_{i:T_i=t} w_i g_t(S_i) \quad \frac{1}{n_j}\sum_{j:T_j=t} w_j g_t(S_i)}{t}
\end{aligned}
\tag{9}
$$

where $\frac{\frac{1}{n_i}\sum_{i:T_i=t} w_i + \frac{1}{n_j}\sum_{j:T_j=t} w_j}{t}$ is the ground truth of MTEF, ε means the noise term, and $\varepsilon \sim 0$ with Gaussian noise. According to the last equation, to reduce the bias of $\widehat{\mathrm{MTEF}}$, we need regularize the term $\sum_k \beta_k \frac{\frac{1}{n_i}\sum_{i:T_i=t} w_i X_{ik} \quad \frac{1}{n_j}\sum_{j:T_j=t} w_j X_{jk}}{t}$ and $\frac{\frac{1}{n_i}\sum_{i:T_i=t} w_i g_t(S_i) \quad \frac{1}{n_j}\sum_{j:T_j=t} w_j g_t(S_i)}{t}$, where the second term has the unknown term $g_T(S_i)$ so that we can only try to reduce the first term. $\frac{\frac{1}{n_i}\sum_{i:T_i=t} w_i X_{ik} \quad \frac{1}{n_j}\sum_{j:T_j=t} w_j X_{jk}}{t}$ means the difference of the k-th confounder between treated and control samples. The parameter β_k represents the confounding bias weight of the k-th confounder, and it is the coefficient of $X_{:k}$. Moreover, because our target is to learn the weight of each variable pair, i.e., between treatment and each confounder, we also need to learn the weight of treatment T. Hence, according to Eq. (8), the confounder weights and treatment weight can be learned by regressing observed outcome Y on confounders X and treatment T. □

Due to the connection between treatment effect estimation with variable decorrelation as analyzed in Section 3.1, we utilize Theorem 1 to reweight the variable weight in variable decorrelation term. When apply the Theorem 1 to GNNs, the confounders X should be $H_{:\setminus j}$ and treatment T is $H_{:j}$, where the embedding H is learned by $G(X; A; \theta_g)$ in Eq. (2). And the variable weights w is equal to the regression coefficients for H, i.e, $\hat{\beta}$ in Eq. (2). Then the Differentiated Variable Decorrelation (DVD) term can be formulated as follows:

$$
\begin{aligned}
\min_w L_{DVD}(H) = & \sum_{j=1}^{p} \Big\| \mathrm{abs}\big(H_{:j}^T w H_{:\setminus j}\big) - \mathrm{abs}\big(H_{:j}^T w H_{:\setminus j}\big)^2 \Big\| \\
& \frac{1}{n}\sum_{i=1}^{n} w_i^2 \quad \Big(\frac{1}{n}\sum_{i=1}^{n} w_i\Big)^2; \\
& \text{s.t. } w \geq 0
\end{aligned}
\tag{10}
$$

where abs means the element-wise absolute value operation, preventing positive and negative values from eliminating. Term

$\frac{1}{n}<_{i\ 1}^{n} w_i^2$ is added to reduce the variance of sample weights to achieve stability, and the formula $_2\ \frac{1}{n}<_{i\ 1}^{n} w_i\ \ 1\ ^2$ avoids all the sample weights to be 0. The term wW0 constrains each sample weight to be non-negative. After variable reweighting, the weighted decorrelation term in Eq. (10) can be rewritten as $<_{j\,j\,k}\ _j^2\ _k^2 \P\P H_{:j}^T\ _w H_{:k}\copyright n\ \ H_{:j}^T w\copyright n\ H_{:k}^T w\copyright n\P\P_2^2$, and the weight for variable pair j and k would be $_j^2\ _k^2$, hence it considers both the weights of treatment and confounder. Then we derive the uniqueness property of w as follows:

Theorem 2 (Uniqueness) If $_1 n\ 9\ p^2\ \ _2, p^2\ 9\ \max\ _1;\ _2\ , \P H_{i;j}\P\ \& c$ and $\P\ _i\P\ \& c$ for some constant c, the solution $\hat{w}$ " $r w\ \P w_i\P\ \& cx$ to minimize Eq. (10) is unique.

Proof. See Appendix A. □

C. Debiased GNN Framework

In this section, we describe the framework of Debiased GNN (DGNN) that incorporates VD/DVD term with GNNs in a seamless way. As analyzed in Section III-B, decorrelating $\hat{A}\ \hat{A}XW^0$ could make parameter estimation of GCN unbiased. However, most GNNs follow a layer-by-layer stacking architecture, and the output embedding of each layer is more easy to obtain in implementing. Since $\hat{A}XW^0$ is the aggregation of the first layer embedding $\hat{A}XW^0$, decorrelating $\hat{A}\ \hat{A}XW^0$ may lack the flexibility that incorporates VD/DVD term with other GNN architectures. Fortunately, we have the following theorem to identify a more flexible way to combine variable decorrelation with GNNs.

Theorem 3. Given p pairwise uncorrelated variables $Z\ Z_1; Z_2;\ ; Z_p$, with a linear aggregation operator $\hat{A}$, the variables of $Y\ \hat{A}Z$ are still pairwise uncorrelated.

Proof. Let $Z\ r Z_1; Z_2;\ ; Z_p x$ be p pairwise uncorrelated variables. $¾Z_i; Z_j\ "\ Z,\ Z_i^{\ 1}; Z_i^{\ 2};\ ; Z_i^{\ n}$ and $Z_j^{\ 1}; Z_j^{\ 2};\ ; Z_j^{\ n}$ are n simple random samples drawn from Z_i and Z_j respectively, and have same distribution with Z_i and Z_j. Given a linear aggregation matrix $\hat{A}\ a_{ij}$, $¾s; v\ "\ 1; 2;\ ; n$, let $Y_i^{\ s}\ <_{k\ 1}^{n} a_{sk} Z_i^{\ k}$ and $Y_j^{\ v}\ <_{l\ 1}^{n} a_{vl} Z_j^{\ l}$, and we have following derivation:

$$\begin{aligned}\mathrm{Cov}\left(Y_i^{\ s}; Y_j^{\ v}\right) &= \mathrm{Cov}\left(\sum_{k\ 1}^{n} a_{sk} Z_i^{\ k}; \sum_{l\ 1}^{n} a_{vl} Z_j^{\ l}\right)\\ &= \sum_{k\ 1}^{n}\sum_{l\ 1}^{n} a_{sk}\, a_{vl}\, \mathrm{Cov}\left(Z_i^{\ k}; Z_j^{\ l}\right)\\ &= \sum_{k\ 1}^{n}\sum_{l\ 1}^{n} a_{sk}\, a_{vl}\ _{ij};\end{aligned}$$

where $_{ij}\ 0$ when $i\ j\ j$, otherwise $_{ij}\ 1$. Therefore, when $i\ j\ j$, we have $\mathrm{Cov}\left(Y_i^{\ s}; Y_j^{\ v}\right)\ 0$ and $\mathrm{Cov}\left(Y_i; Y_j\right)\ 0$. Extended the conclusion to multiple variable, $Y\ Y_1; Y_2;\ ; Y_n$ are pairwise uncorrelated. □

The theorem indicates that if the variables of embeddings Z are uncorrelated, after any form of linear neighborhood aggregation $\hat{A}$, e.g., average, attention or sum, the variables of transformed embeddings Y would be also uncorrelated. Therefore, decorrelating $\hat{A}XW^0$ can also reduce the estimation bias. For a K layers of GNN, we can directly decorrelate the output of $K\ \ 1$ -th layer, i.e., $\hat{A}\ \ \hat{A}XW^0\ \ W^{K\ 2}$ for a K layers of GCN.

The previous analysis finds a flexible way to incorporate VD/DVD term with GNNs, however, recall that we analyze GNNs based on the least-squares loss in Eq. (3), and most existing GNNs are designed for the classification task. Therefore, in the following, we analyze that the previous conclusions are still applicable in classification. We consider the cases that the softmax layer is used as the output layer of GNNs and loss is the cross-entropy error function. We use the Newton-Raphson update rule [17] to bridge the gap between linear regression and multi-classification. According to the Newton-Raphson update rule, the update formula for transformation matrix $W^{K\ 1}$ of the last layer of GCN can be derived:

$$\begin{aligned}W_{:j}^{\ new} &= W_{:j}^{\ old}\ \ \left(H^T R H\right)^{\ 1} H^T\left(H W_{:j}^{\ old}\ \ Y_{:j}\right)\\ &= \left(H^T R H\right)^{\ 1}\left(r H^T R H W_{:j}^{\ old}\ \ H^T\left(H W_{:j}^{\ old}\ \ Y_{:j}\right)\right) x\\ &= \left(H^T R H\right)^{\ 1} H^T R z;\end{aligned} \tag{11}$$

where $R_{kj}\ <_{n\ 1}^{N} H_{n} W_{:k}^{\ old} I_{kj}\ \ H_{n} W_{:j}^{\ old}$ is a weighing matrix and I_{kj} is the element of the identity matrix, and $z\ \ H W_{:j}^{\ old}\ \ R^{\ 1}\left(Y_{:j}\ \ W_{:j} H\right)$ is an effective target value. Eq. (11) takes the form of a set of normal equations for a weighted least-squares problem. As the weighing matrix R is not constant but depends on the parameter vector $W_{:j}^{\ old}$, we must apply the normal equations iteratively. Each iteration uses the last iteration weight vector $W_{:j}^{\ old}$ to calculate a revised weighing matrix R and regresses the target value z with $W_{:j}^{\ new}$. Therefore, the variable decorrelation can also be applied to the GNNs with softmax classifier to reduce the estimation bias in each iteration. Note that according to update formula Eq. (11), we should calculate the inverse matrix $\left(H^T R H\right)^{\ 1}$ in each iteration, which requires high computation. In practice, we use gradient descent methods to approximate Newton-Raphson update rule and it works well in experiments.

Fig. 2(b) is the framework of GNN-DVD, where we input the labeled nodes' embeddings $H^{K\ 1}$ into the regularizer $L_{DVD}\left(H^{K\ 1}\right)$. As GCN has the formula $\mathrm{softmax}\left(\hat{A}H^{K\ 1} W^{K\ 1}\right)$, the variable weights of $H^{K\ 1}$ used for differentiating $L_{DVD}\left(H^{K\ 1}\right)$ can be computed from

$$\mathrm{Var}\left(W^{K\ 1}; \mathrm{axis}\ \ 1\right) \tag{12}$$

where $\mathrm{Var}\ ; \mathrm{axis}\ \ 1$ refers to calculating the variance of each row of some matrix and it reflects each variable's weight for classification which is similar to the regression coefficients. Note that when incorporating VD term with GNNs, we do not need compute the variable weights. Then the sample weights learned by DVD term have the ability to remove the correlation in $H^{K\ 1}$. Inspired by sample weighting methods [18], we propose to use this sample weights to reweight softmax loss:

$$\min L_G\ =\ \sum_{i"\,Y_L} w_i\ \ \ln\ q\left(H_i^{\ K}\ \ Y_i\right); \tag{13}$$

where q is the softmax function, Y_L is the set of labeled node indices and θ is the set of parameters of GCN. After reweighting nodes by these weights, we can create a pseudo-population where the biases in node neighborhood are effectively reduced, with which the off-the-shelf GNN models can achieve more accurate prediction under agnostic environments. The whole algorithm is summarized in Algorithm 1.

Algorithm 1: GNN-DVD Algorithm

Input : Training graph $G_{train}(A, X, Y_X)$, and indices of labeled nodes Y_L; Max iteration $maxIter$

Output : GNN parameter θ and sample weights w

Initialization : Let $w = 1/n$ and initialize sample weights w with 1; Initialize GNN's parameters θ with random uniform distribution; Iteration $t = 0$

```
while not converged or t < maxIter do
    Optimize θ^t to minimize L_G via Eq. (13);
    Calculate variable weights w^t from W^{K-1} via Eq. (12);
    Optimize w^t to minimize L_DVD(H^{K-1}) via Eq. (10);
    t = t + 1;
end
```

Return: θ and $w = 1/n$

To optimize our GNN-DVD algorithm, we propose an iterative method. Firstly, we let $w = 1/n$ to ensure non-negativity of w and initialize sample weight $w_i = 1$ for each sample i and GNN's parameters θ with random uniform distribution. Once the initial values are given, in each iteration, we fix the sample weights w and update the GNN's parameters θ by L_G with gradient descent, then compute the confounder weights w from the linear transform matrix W^{K-1}. With θ and w fixing the GNN's parameters, we update the sample weights w with gradient descent to minimize $L_{DVD}(H^{K-1})$. We iteratively update the sample weights w and GNN's parameters θ until L_G converges.

D. Extension to GAT

We can easily incorporate the VD/DVD term with other GNNs. We combine them with GAT and more extensions leave as future work. GAT utilizes an attention mechanism to aggregate neighbor information. It also follows the linear aggregation and transformation steps. Similar to GCN, the hidden embedding H^{K-1} is the input of VD/DVD term, and the variable weights w are calculated from the transformation matrix W^{K-1} and the sample weights w are used to reweight the softmax loss. Note that the original paper utilizes the same transformation matrix W^{K-1} for transforming embedding and learning attention values. Because w means the importance of each variable for classification, and it should be calculated from transformation matrix W^{K-1} for transforming embedding, hence we use separate matrices for transforming embedding and learning attention values, respectively. This modification does not change the performance of GAT in experiments.

E. Complexity Analysis

Compared with base models (e.g., GCN and GAT), the main incremental time cost is the complexity from VD/DVD term. For a training graph with n labeled nodes, we analyze the time complexity of the VD/DVD term in each iteration. For calculating the VD loss, its complexity is $O(np^2)$, where p is the dimension of embeddings. And for DVD loss, its complexity is the same as VD, as the complexity of calculating variable weights w is $O(np)$, which is relatively small comparing with $O(np^2)$. For updating w, the complexity is dominated by the step of calculating the partial gradients of the function $L_{DVD}(H)$ with respect to variable w. The complexity of $\frac{\partial L_{DVD}(H)}{\partial w}$ is $O(np^2)$. In total, the complexity of each iteration for VD/DVD term in Algorithm 1 is $O(np^2)$. And it is quite smaller than the base models (e.g., the complexity of GCN is $O(Ecp^2)$, where E is the number of edges and c is the dimension of input node features).

F. Discussion

In our paper, we propose to integrate two decorrelation terms (i.e., VD/DVD term) with GNN models to eliminate the estimation bias. Here we discuss the advantages and disadvantages of these two terms. VD term aggressively decorrelates all the variables learned by GNNs, however, it theoretically requires a large number of samples to achieve this goal. To overcome this dilemma, the DVD term is proposed to differentiate the variable weights in the VD term, aiming to remove the most unexpected correlation. However, due to the existence of unknown term αS in Eq. (8), introducing more parameters to optimize may increase the instability of the model. Hence, when the number of labeled samples is large, performing GNN-VD may induce more stable results.

IV. Experiments

A. Datasets

Here, we validate the effectiveness of our methods on node classification with two kinds of selection bias, i.e., label selection bias and small sample selection bias. For label selection bias, we employ three widely used graph datasets: Cora, Citeseer and Pubmed [7]. As in Section II-A, we get three biased degrees as well as the original unbiased labeled nodes for each dataset. For small sample selection bias, we conduct the experiments on NELL dataset [19] where each class only has at most 1/5/10 labeled nodes for training. Due to the large scale of this dataset, the test nodes are easily to have distribution shifts from training nodes. The details of the datasets are summarized in Table II.

B. Baselines

We compare our proposed framework with several related baselines:

- Base models: GCN [2] and GAT [3] are classical GNN methods. We utilize them as the base models in our

TABLE II: Dataset statistics

Dataset	Type	Nodes	Edges	Classes	Features	Bias degree ( )	Bias type
Cora	Citation network	2,708	5,429	7	1,433	0.7/0.8/0.9	Label selection bias
Citeseer	Citation network	3,327	4,732	6	3,703	0.7/0.8/0.9	Label selection bias
Pubmed	Citation network	19,717	44,338	3	500	0.7/0.8/0.9	Label selection bias
NELL	Knowledge graph	65,755	266,144	210	5,414	1/5/10 labeled nodes per class	Small sample selection bias

framework, so they are the most related baselines to validate the effectiveness of the proposed framework.

- GNM-GCN/GAT [20]: A GNN method which considers unbalanced label selection bias problem in transductive setting. They also utilize GCN/GAT as their base models.
- Chebyshev [2]: It is a GCN-based method utilizing third-order Chebyshev lters.
- SGC [8]: It is a simpli ed GCN-based method, which reduces the excess complexity through successively removing nonlinearities and collapsing weight matrices between consecutive layers.
- APPNP [21]: It is one of the state-of-the-art GNN methods that combines PageRank with GCN.
- Planetoid [10]: It is a classical semi-supervised graph embedding method. We use its inductive variant.
- MLP: It is a two-layer multilayer perceptron trained on the labeled nodes with only node features as input.
- DGNN: It is the debiased GNN framework proposed in this paper. We incorporate the VD/DVD term with GCN/GAT under our proposed framework called GCN/GAT-VD/DVD.

C. Experimental Setup

As the Section II-A has described, for all datasets, to simulate the agnostic selection bias scenario, we rst follow the inductive setting in [8] that masks the validation and test nodes in the training phase and validation and test with the whole graph so that the test nodes will be agnostic. For GCN and GAT, we utilize the same two-layer architecture as their original paper [2], [3]. We use the following sets of hyperparameters for GCN on Cora, Citeseer, Pubmed: 0.5 (dropout rate), $5e{-}4$ (L2 regularization) and 32 (number of hidden units); and for NELL: 0.1 (dropout rate), $1e{-}5$ (L2 regularization) and 64 (number of hidden units). For GAT on Cora, Citeseer, we use: 8 (rst layer attention heads), 8 (features each head), 1 (second layer attention head), 0.6 (dropout), $5e{-}4$ (L2 regularization); and for Pubmed: 8 (second layer attention head), $1e{-}3$ (L2 regularization), other parameters are the same as Cora and Citeseer. To fair comparison, the GNN part of our model uses the same architecture and hyper-parameters with the base model, and we grid search $_1$ and $_2$ from $\{0.01, 0.1, 1, 10, 100\}$. For other baselines, we use the optimal hyper-parameters in the original literatures on each dataset. For all the experiments, we run each model 10 times with different random seeds and report its average Accuracy results.

D. Results on Label Selection Bias Datasets

The results are given in Table III, and we have the following observations. First, the proposed models (i.e., GCN/GAT with VD/DVD terms) always achieve the best performances in most cases, which well demonstrates that the effectiveness of our proposed debiased GNN framework. Second, comparing with base models, our proposed models all achieve up to 17.0% performance improvements, and gain larger improvements under heavier bias scenarios. Since the major difference between our model with base models is the VD/DVD regularizer, we can safely attribute the signi cant improvements to the effective decorrelation term and its seamless joint with GNN models. Third, GCN/GAT-DVD achieves better results than GCN/GAT-VD in most cases. It validates the importance and effectiveness of differentiating variables' weights in the semi-supervised setting. Moreover, our model still outperforms baselines in the unbiased setting. In real applications, it is hard to control the collection process without any distribution shift from the training set to the test set [23]. Therefore, the problem we study is ubiquitous in reality and our method is effective in most scenarios.

E. Results on Small Sample Selection Bias Datasets

As NELL is a large-scale graph, we cannot run GAT on a single GPU with 16GB memory. We only perform GCN-VD/DVD and compare with representative methods which can perform on this dataset. The results are shown in Table V. First, GCN-VD/DVD achieves signi cant improvements over GCN. It indicates that selection bias could be induced by a small number of labeled nodes and our proposed method relieve the estimation bias. Moreover, with fewer labeled nodes, i.e., larger selection bias, our methods achieve larger improvements over base models. It further validates our method is an effective method against heavy bias. Moreover, GCN-DVD further improves GCN-VD with a large margin on NELL-1 dataset. It means that decorrelating all the variable pairs equally is suboptimal, and our differentiated strategy is effective when labeled nodes are scarce. With the number of labeled nodes increases, it may not necessary to differentiate the variable weights, but GCN-DVD achieves competitive results with GCN-VD and still outperforms the base model with a clear margin.

F. Sample Weight Analysis

Here we analyze the effect of sample weights. We compute the amount of correlation in the labeled nodes' embeddings $H_{:,K-1}$ learned by standard GCN and the weighted embeddings of the same layer learned by GCN-DVD. Note that, the weights are the last iteration of sample weights of GCN-DVD. Following [24], [25], the amount of correlation of GCN and GCN-DVD is measured by Frobenius norm of cross-covariance matrix $\|C\|_F^2$ computed from variables of $H_{:,K-1}$ and weighted $H_{:,K-1}$ respectively, where C_{ij} represents the

TABLE III: Performance of three citation networks. The `*' indicates the best results of the baselines. Best results of all methods are indicated in bold. `% gain over GCN/GAT' means the improvement percent of GCN/GAT-DVD against GCN/GAT.

Method	Cora				Citeseer				Pubmed			
	Unbiased	Light	Medium	Heavy	Unbiased	Light	Medium	Heavy	Unbiased	Light	Medium	Heavy
MLP	0.5296	0.5624	0.5197	0.5087	0.5438	0.4532	0.3757	0.3893	0.6914	0.6852	0.6620	0.6378
Planetoid [10]	0.6650	0.5890	0.5240	0.5180	0.6720	0.5160	0.5140	0.4880	0.744	0.7160	0.6770	0.6680
Chebyshev [22]	0.7407	0.7116	0.7006	0.6809	0.7232	0.6542	0.6276	0.5920	0.7450	0.7358	0.6862	0.6732
SGC [8]	0.779	0.7800	0.7800	0.7530	0.724	0.6780	0.6730	0.6200	0.781	0.7880	0.7560	0.6800
APPNP [21]	0.8132	0.7913	0.7689	0.7629	0.6862	0.6478	0.6052	0.5903	0.7731	0.7639	0.7369	0.6862
GNM-GCN [20]	0.7594	0.7423	0.7531	0.7196	0.6054	0.5793	0.5717	0.5125	0.7654	0.7552	0.7381	0.7072
GNM-GAT [20]	0.7976	0.7875	0.7638	0.7404	0.6832	0.6524	0.6487	0.5865	0.7666	0.7438	0.7568	0.6891
GCN [2]	0.7909	0.7851	0.7775	0.7422	0.7075	0.6786	0.5952	0.5551	0.7845	0.7673	0.7545	0.7247
GCN-VD	0.7980	0.7951	0.7855	0.7522	0.7122	0.6844	0.6676	0.6408	0.7888	0.7727	0.7729	0.7399
GCN-DVD	0.7951	0.7959	0.7885	0.7555	0.7128	0.6908	0.6769	0.6496	0.7874	0.7741	0.7746	0.7542
% gain over GCN	0.53%	1.38%	1.41%	1.79%	0.75%	1.8%	14.2%	17.0%	0.37%	0.89%	2.67%	4.07%
GAT [3]	0.8100	0.8067	0.8019	0.7578	0.7224	0.7033	0.6683	0.6475	0.7714	0.7665	0.7579	0.7068
GAT-VD	0.8133	0.8146	0.8079	0.7708	0.7288	0.7149	0.6833	0.6611	0.7732	0.7783	0.7689	0.7149
GAT-DVD	0.8139	0.8179	0.8119	0.7694	0.7294	0.7172	0.6825	0.6627	0.7735	0.7788	0.7723	0.7210
% gain over GAT	0.48%	1.39%	1.26%	1.53%	0.97%	1.97%	2.12%	2.34%	0.27%	1.6%	1.9%	2.0%

TABLE IV: Performance of NELL. The `*' indicates the best results of the baselines. Best results of all methods are indicated in bold. `Improvement' means the improvement percent of GCN-VD/DVD (selected better results) against GCN.

Dataset	MLP	Planetoid	SGC	GCN	GCN-VD	GCN-DVD	Improvement
NELL-1	0.2385	0.3901	0.4128	0.4416	0.4652	0.4734	7.2%
NELL-5	0.4938	0.3519	0.6295	0.7030	0.7424	0.7361	5.6%
NELL-10	0.5838	0.5149	0.6275	0.7615	0.7734	0.7727	1.6%

(a) Cora (b) Citeseer (c) Pubmed

Fig. 3: Embedding correlation analysis on unweighted and weighted GCN.

covariance between pairwise variable i and j and the main diagonal of C is set as zero vector. Figure 3 shows the amount of correlation in unweighted and weighted embeddings, and we observe that the embeddings' correlations in all datasets are reduced, indicating that the weights learned by GCN-DVD can reduce the correlation between embedded variables. Moreover, as it is hard to reduce the correlation to zero, the necessity of differentiating variables' weights is further validated.

G. Parameter sensitivity

We study the sensitiveness of parameters and report the results of GCN-DVD on three citation networks in Fig. 4-6. The experimental results show that GCN-DVD is relatively stable to $_1$ and $_2$ with wide ranges in most cases, indicating the robustness of our model.

H. Training time per epoch

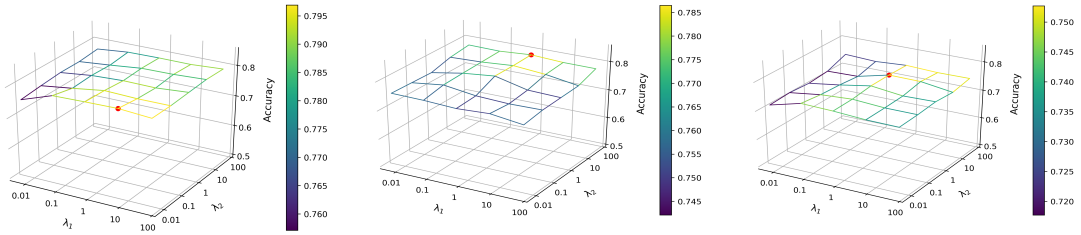
We report the results for the mean training time of GCN and GCN-DVD per epoch (forward pass, cross-entropy calculation, backward pass) for 200 epochs on Cora, Citeseer and Pubmed datasets, measured in seconds wall-clock time. These methods are performed on a RTX 3090 GPU Card. As we can see, the training time of GCN-DVD term has the same order of magnitude with GCN. More importantly, the training time of the DVD term will not be influenced by the base model we choose, i.e., when the base model is GAT, the running time of DVD term will not change.

TABLE V: The training time per epoch.

	Cora		Citeseer		Pubmed	
GCN	1:29	10^{-2}	2:00	10^{-2}	1:11	10^{-1}
GCN-DVD	6:19	10^{-2}	8:46	10^{-2}	2:42	10^{-1}

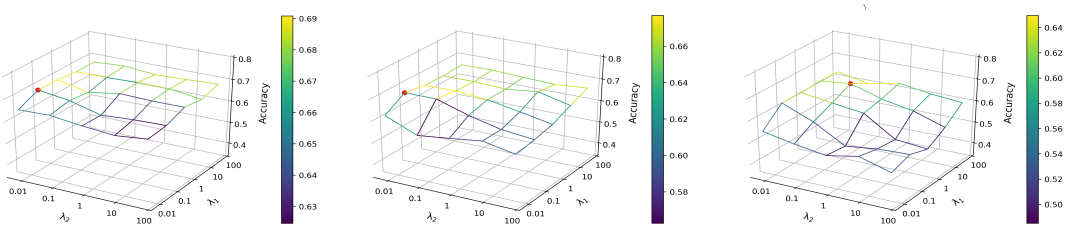
V. Related Works

In the past few years, Graph Neural Networks (GNNs) [1], [2], [3], [21], [26], [27], [28], [29], [30] have become the major technology to capture patterns encoded in the graph due to its powerful representation capacity. Recently, KPGNN [31] applies GNNs to the social event detection task by preserving the incremental knowledge emerging in social data. MRFasGCN [32] integrates GCN with a Markov Random Fields (MRF) model to deal with the semi-supervised community detection problem. Not only pursuing the performance of GNNs



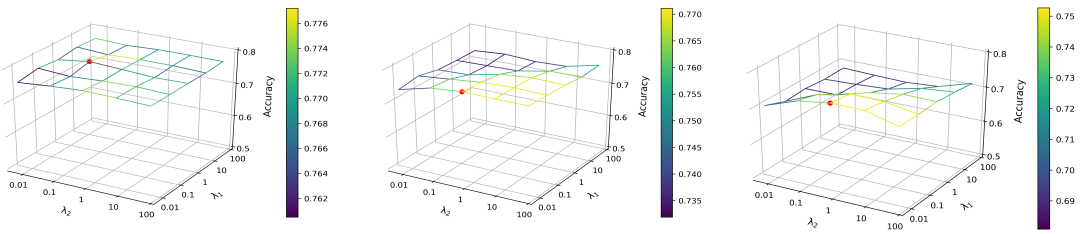
(a) Light (b) Medium (c) Heavy

Fig. 4: Accuracy of GCN-DVD with different $_1$ and $_2$ on different biased Cora datasets.



(a) Light (b) Medium (c) Heavy

Fig. 5: Accuracy of GCN-DVD with different $_1$ and $_2$ on different biased Citeseer datasets.



(a) Light (b) Medium (c) Heavy

Fig. 6: Accuracy of GCN-DVD with different $_1$ and $_2$ on different biased Pubmed datasets.

on clean data, [33] proposes an exploratory adversarial attack method, called EpoAtk, to test whether existing GNNs are robust with adversarial perturbations on graphs. Although the current GNNs have achieved great success, when applied to the inductive setting, they all assume that training nodes and test nodes follow the same distribution. However, this assumption does not always hold in real applications. GNM [20] first pays attention to the label selection problem on graph learning, and it learns an IPW estimator to estimate the probability of each node to be selected and uses this probability to reweight the labeled nodes. However, it heavily relies on the accuracy of the IPW estimator, which depends on the label assignment distribution of the whole graph, hence it is more suitable for the transductive setting.

To enhance the stability in unseen varied distributions, some literatures [5], [34] have revealed the connection between correlation and prediction stability under model misspecification. Moreover, a kind of literatures [35], [36], [37] have studied the problem of removing the features correlation effect in neural networks, which brings great benefits for deep neural networks. However, these methods are built on simple regressions or regular neural networks such as CNNs, but GNNs have more complex architectures and properties needed to be considered. We also notice that [6] propose a differentiated variable decorrelation term for linear regression. However, this decorrelation term requires multiple environments with different correlations between stable variables and unstable variables available in the training stage while our method does not require this prior knowledge.

VI. Conclusion

In this paper, we investigate a general and practical problem: learning GNNs with agnostic label selection bias. The selection bias will inevitably cause the GNNs to learn the biased correlation between aggregation mode and class label and make the prediction unstable. We propose a novel debiased GNN framework, which combines the decorrelation technique with GNNs in a unified framework. Extensive experiments well demonstrate the effectiveness and flexibility of DGNN.

ACKNOWLEDGMENT

This work is partially supported by the National Natural Science Foundation of China (No. U20B2045, 62192784, 62172052, 61772082, 62002029) and the Fundamental Research Funds for the Central Universities (No. 2021RC28). Kun Kuang is supported in part by National Natural Science Foundation of China (No. 62006207), Young Elite Scientists Sponsorship Program by CAST and Zhejiang Province Natural Science Foundation (No. LQ21F020020). Shaohua Fan is supported by BUPT Excellent Ph.D. Students Foundation (No. CX2021311) and China Scholarship Council. We thank Dr. Tianchi Yang for discussion on the proof of Theorem 3.

APPENDIX A
PROOF OF THEOREM 2

$$\hat{\mathbf{w}} \quad \arg\min_{\mathbf{w}} \sum_{j=1}^{p} \quad ^{\mathsf{T}} \text{abs}\ \mathbf{H}_{:j}^{\mathsf{T}} \quad \mathbf{w}\mathbf{H}_{:\ j}\copyright n \quad \mathbf{H}_{:j}^{\mathsf{T}}\mathbf{w}\copyright n \quad \mathbf{H}_{:\ j}^{\mathsf{T}}\mathbf{w}\copyright n \quad ^2 \quad \frac{1}{n}\sum_{i=1}^{n} \mathbf{w}_i^2 \quad _2 \frac{1}{n}\sum_{i=1}^{n} \mathbf{w}_i \quad 1 \quad ^2 \tag{14}$$

Proof. For simplicity, we denote $L_1 \quad \sum_{j=1}^{p} \quad ^{\mathsf{T}} \text{abs}\ \mathbf{H}_{:j}^{\mathsf{T}} \quad \mathbf{w}\mathbf{H}_{:\ j}\copyright n \quad \mathbf{H}_{:j}^{\mathsf{T}}\mathbf{w}\copyright n \quad \mathbf{H}_{:\ j}^{\mathsf{T}}\mathbf{w}\copyright n \quad ^2$, $L_2 \quad \frac{1}{n}\sum_{i=1}^{n} \mathbf{w}_i^2$, $L_3 \quad \frac{1}{n}\sum_{i=1}^{n} \mathbf{w}_i \quad 1 \quad ^2$ and $F\ \mathbf{w} \quad L_1 \quad _1 L_1 \quad _2 L_2$. We first calculate the Hessian matrix of $F\ \mathbf{w}$, denoted as $\mathbf{H}_e$, to prove the uniqueness of the optimal solution $\hat{\mathbf{w}}$, as follows:

$$\mathbf{H}_e \quad \frac{@^2 L_1}{@\mathbf{w}^2} \quad _1 \frac{@^2 L_2}{@\mathbf{w}^2} \quad _2 \frac{@^2 L_3}{@\mathbf{w}^2}$$

For the term L_1, we can rewrite it as:

$$\begin{aligned} L_1 \quad &= \sum_{j\,j\,k}^{2\ \ 2} \frac{1}{n}\sum_{i=1}^{n} \mathbf{H}_{i;j}\mathbf{H}_{i;k}\mathbf{w}_i \quad \frac{1}{n}\sum_{i=1}^{n} \mathbf{H}_{i;j}\mathbf{w}_i \quad \frac{1}{n}\sum_{i=1}^{n} \mathbf{H}_{i;k}\mathbf{w}_i \quad ^2 \\ &= \sum_{j\,j\,k}^{2\ \ 2} \frac{1}{n}\sum_{i=1}^{n} \mathbf{H}_{i;j}\mathbf{H}_{i;k}\mathbf{w}_i \quad ^2 \quad \frac{2}{n}\sum_{i=1}^{n} \mathbf{H}_{i;j}\mathbf{H}_{i;k}\mathbf{w}_i \quad \frac{1}{n}\sum_{i=1}^{n} \mathbf{H}_{i;j}\mathbf{w}_i \quad \frac{1}{n}\sum_{i=1}^{n} \mathbf{H}_{i;k}\mathbf{w}_i \\ &\quad \frac{1}{n}\sum_{i=1}^{n} \mathbf{H}_{i;j}\mathbf{w}_i \quad \frac{1}{n}\sum_{i=1}^{n} \mathbf{H}_{i;k}\mathbf{w}_i \quad ^2 \end{aligned}$$

And when $\P\mathbf{H}_{i;j}\P \ \& \ c$, for any variable j and k, and $\P\mathbf{w}_i\P \ \& \ c$, we have $\frac{@^2}{@\mathbf{w}^2} \quad \frac{1}{n}\sum_{i=1}^{n} \mathbf{H}_{i;j}\mathbf{H}_{i;k}\mathbf{w}_i \quad ^2 \quad O\ \frac{1}{n^2}$, $\frac{@^2}{@\mathbf{w}^2} \quad \frac{1}{n}\sum_{i=1}^{n} \mathbf{H}_{i;j}\mathbf{w}_i \quad \frac{1}{n}\sum_{i=1}^{n} \mathbf{H}_{i;k}\mathbf{w}_i \quad O\ \frac{1}{n^2}$ and $\frac{@^2}{@\mathbf{w}^2} \quad \frac{2}{n}\sum_{i=1}^{n} \mathbf{H}_{i;j}\mathbf{H}_{i;k}\mathbf{w}_i \quad \frac{1}{n}\sum_{i=1}^{n} \mathbf{H}_{i;j}\mathbf{w}_i \quad \frac{1}{n}\sum_{i=1}^{n} \mathbf{H}_{i;k}\mathbf{w}_i \quad O\ \frac{1}{n^2}$. Then with $\P\ _i\P \ \& \ c$, we have $\sum_{i\ \ k}^{2\ \ 2} \frac{@^2}{@\mathbf{w}^2} \quad \frac{1}{n}\sum_{i=1}^{n} \mathbf{H}_{i;j}\mathbf{H}_{i;k}\mathbf{w}_i \quad \frac{1}{n}\sum_{i=1}^{n} \mathbf{H}_{i;j}\mathbf{w}_i \quad \frac{1}{n}\sum_{i=1}^{n} \mathbf{H}_{i;k}\mathbf{w}_i \quad ^2 \quad O\ \frac{1}{n^2}$. L_1 is sum of $p\ p \quad 1$ such terms. Then we have

$$\frac{@^2 L_1}{@\mathbf{w}^2} \quad O\ \frac{p^2}{n^2}\ :$$

With some algebras, we can also have

$$\frac{@^2 L_2}{@\mathbf{w}^2} \quad \frac{1}{n}\mathbf{I}; \frac{@^2 L_3}{@\mathbf{w}^2} \quad \frac{1}{n^2}\mathbf{1}\mathbf{1}^{\mathsf{T}};$$

thus,

$$\mathbf{H}_e \quad O\ \frac{p^2}{n^2} \quad \frac{1}{n}\mathbf{I} \quad \frac{2}{n^2}\mathbf{1}\mathbf{1}^{\mathsf{T}} \quad \frac{1}{n}\mathbf{I} \quad O\ \frac{p^2}{n^2} \quad ^2\ :$$

Therefore, if $\frac{1}{n} \ 9 \ \frac{p^2 \quad ^2}{n^2}$, equivalent to $_1 n \ 9 \ p^2 \quad _2$, $\mathbf{H}_e$ is an almost diagonal matrix. Hence, $\mathbf{H}_e$ is positive definite [38]. Then the function $F\ \mathbf{w}$ is convex on $C \quad \mathrm{r}\mathbf{w} \quad \P\mathbf{w}_i\P \ \& \ cx$, and has unique optimal solution $\hat{\mathbf{w}}$.

Moreover, because L_1 is our major decorrelation term, we hope L_1 to dominate the terms $_1 L_2$ and $_2 L_3$. On C, we have $L_1 \quad O\ 1\ , \ L_2 \quad O\ 1\ ,$ and $\sum_{i\ \ k}^{2\ \ 2} \frac{1}{n}\sum_{i=1}^{n} \mathbf{H}_{i;j}\mathbf{H}_{i;k}\mathbf{w}_i \quad \frac{1}{n}\sum_{i=1}^{n} \mathbf{H}_{i;j}\mathbf{w}_i \quad \frac{1}{n}\sum_{i=1}^{n} \mathbf{H}_{i;k}\mathbf{w}_i \quad ^2 \quad O\ 1$. Thus $L_1 \quad O\ p^2$. When $p^2 \ 9 \ \max\ _1; \ _2\ , \ L_1$ will dominate the regularization terms L_2 and L_3. □

REFERENCES

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

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

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

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

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

[6] Z. Shen, P. Cui, J. Liu, T. Zhang, B. Li, and Z. Chen, "Stable learning via differentiated variable decorrelation," in *KDD*, 2020, pp. 2185–2193.

[7] 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.

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

[9] B. Zadrozny, "Learning and evaluating classifiers under sample selection bias," in *ICML*, 2004, p. 114.

[10] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," in *ICML*, 2016.

[11] M. Nurhonen and S. Puntanen, "A property of partitioned generalized regression," *Communications in statistics-theory and methods*, vol. 21, no. 6, pp. 1579–1583, 1992.

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

[13] K. Kuang, L. Li, Z. Geng, L. Xu, K. Zhang, B. Liao, H. Huang, P. Ding, W. Miao, and Z. Jiang, "Causal inference," *Engineering*, vol. 6, no. 3, pp. 253–263, 2020.

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

[15] K. Kuang, P. Cui, H. Zou, B. Li, J. Tao, F. Wu, and S. Yang, "Data-driven variable decomposition for treatment effect estimation," *TKDE*, 2020.

[16] N. Kreif, R. Grieve, I. Díaz, and D. Harrison, "Evaluation of the effect of a continuous treatment: a machine learning approach with an application to treatment for traumatic brain injury," *Health economics*, vol. 24, no. 9, pp. 1213–1228, 2015.

[17] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

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

[19] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka, and T. M. Mitchell, "Toward an architecture for never-ending language learning," in *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[20] F. Zhou, T. Li, H. Zhou, H. Zhu, and J. Ye, "Graph-based semi-supervised learning with non-ignorable non-response," in *NeurIPS*, 2021.