UD-GNN: Uncertainty-aware Debiased Training on Semi-Homophilous Graphs

Yang Liu[†]
Institute of Computing Technology, CAS
University of Chinese Academy of Sciences
Beijing, China
liuyang520ict@gmail.com

Xiang Ao[*†]
Institute of Computing Technology, CAS
University of Chinese Academy of Sciences
Beijing, China
aoxiang@ict.ac.cn

Fuli Feng
University of Science and Technology of China
Hefei, Anhui, China
fulifeng93@gmail.com

Qing He[*†]
Institute of Computing Technology, CAS
University of Chinese Academy of Sciences
Beijing, China
heqing@ict.ac.cn

ABSTRACT

Recent studies on Graph Neural Networks (GNNs) point out that most GNNs depend on the homophily assumption but fail to generalize to graphs with heterophily where dissimilar nodes connect. The concept of homophily or heterophily defined previously is a global measurement of the whole graph and cannot describe the local connectivity of a node. From the node-level perspective, we find that real-world graph structures exhibit a mixture of homophily and heterophily, which refers to the co-existence of both homophilous and heterophilous nodes. Under such a mixture, we reveal that GNNs are severely biased towards homophilous nodes, suffering a sharp performance drop on heterophilous nodes. To mitigate the bias issue, we explore an Uncertainty-aware Debiasing (UD) framework, which retains the knowledge of the biased model on certain nodes and compensates for the nodes with high uncertainty. In particular, UD estimates the uncertainty of the GNN output to recognize heterophilous nodes. UD then trains a debiased GNN by pruning the biased parameters with certain nodes and retraining the pruned parameters on nodes with high uncertainty. We apply UD on both homophilous GNNs (GCN and GAT) and heterophilous GNNs (Mixhop and GPR-GNN) and conduct extensive experiments on synthetic and benchmark datasets, where the debiased model consistently performs better and narrows the performance gap between homophilous and heterophilous nodes[1].

CCS CONCEPTS

• **Mathematics of computing** → *Graph algorithms*; • **Computing methodologies** → *Neural networks*.

[*]Corresponding author.
[†]Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS). Xiang Ao is also at Institute of Intelligent Computing Technology, Suzhou, China.
[1]Code is available at https://github.com/PonderLY/UD-GNN

KDD '22, August 14–18, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9385-0/22/08.
https://doi.org/10.1145/3534678.3539483

KEYWORDS

Graph Neural Network, Homophily, Uncertainty, Debiased Training

ACM Reference Format:
Yang Liu, Xiang Ao, Fuli Feng, and Qing He. 2022. UD-GNN: Uncertainty-aware Debiased Training on Semi-Homophilous Graphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3534678.3539483

1 INTRODUCTION

Graph Neural Networks (GNNs) have become the promising solution for various graph-based learning tasks, such as social recommendation [12, 14], fraud detection [13, 22, 39], and drug discovery [21, 29]. The main idea of GNNs is to enhance the target node representation by propagating neighbor representations over the graph structure. Recent studies [40, 41] point out that most GNNs work well under the homophily assumption but fails on heterophilous graphs where dissimilar nodes connect. To determine a graph to be homophilous or heterophilous, the concept of *homophily ratio* is defined as the fraction of edges connecting nodes with the same labels [23]. Graphs with higher homophily ratios are homophilous and those with lower ratios are non-homophilous.

However, the homophily ratio is a graph-level measure of the whole graph and cannot describe the local connectivity of each node. We define *node-level homophily ratio* as the fraction of neighbors with the same label as the node and determine whether a node is homophilous or heterophilous based on a threshold[2] of the ratio.

From the node-level perspective, non-homophilous graphs can be further categorized into semi-homophilous graphs and heterophilous graphs. Semi-homophilous graphs indicate that both homophilous and heterophilous nodes co-exist in the same graph. For instance, in social graphs, people make friends with not only those who share similar interests but also those with different backgrounds. In citation graphs, interdisciplinary papers cite papers from not only the same research areas but also other related areas. To illustrate this categorization with statistics, we calculate the node-level homophily ratio of nine graphs from several benchmarks [20, 25] and divide nodes into four groups, which are colored

[2]In this work, we simply set the threshold to be 0.5.

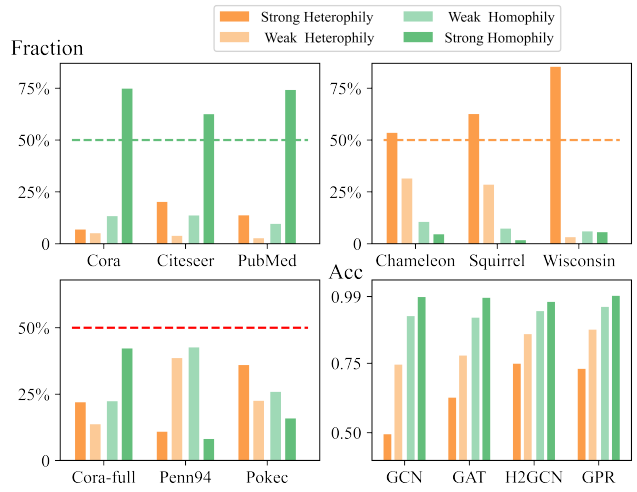


Figure 1: The nodes are categorized into four groups according to node-level homophily ratio: strong heterophily (0-0.25), weak heterophily (0.25-0.5), weak homophily (0.5-0.75), and strong homophily (0.75-1). The first three figures demonstrate the distribution of node groups in the nine graph datasets and the last figure shows the group-level accuracies of the four GNNs on the Cora-full dataset.

in Figure 1. *Cora, Citeseer*, and *PubMed* are typically homophilous graphs where the strong homophily group contains more than 50% nodes. *Chameleon, Squirrel*, and *Wisconsin* are heterophilous graphs with more than half nodes belonging to the strong heterophily group. Differently, in *Penn94, Cora-full*, and *Ogbn-arxiv*, neither strong homophily nor strong heterophily makes up over 50% of the whole set. They contain nodes mixing with different levels of homophily ratio and are called semi-homophilous graphs.

The performance of GNNs trained on semi-homophilous graphs tends to be biased towards homophilous nodes. Empirical evidence is shown in Figure 1, where we conduct a group-wise evaluation of four representative GNNs on Cora-full w.r.t. the level of homophily ratio. For strong homophilous nodes (ratio larger than 0.75), the accuracy is close to 0.99. However, for strong heterophilous nodes (ratio smaller than 0.25), the accuracy ranges from 0.48 to 0.74. For both homophilous GNNs (GCN and GAT) and heterophilous GNNs (H2GCN and GPR-GNN), the performance gap exists with a range from 0.25 to 0.5. The significant performance gap between homophilous and heterophilous nodes will lead to serious issues in real-world applications, such as unstable online performances and unfair services across users.

It is however non-trivial to mitigate such bias in GNNs. The first challenge lies in the difficulty of distinguishing homophilous and heterophilous nodes due to the unavailable labels. For transductive node classification, only part of node labels are available on the whole graph during the training phase. As a result, the node-level homophily ratio should be carefully considered. As demonstrated in Figure 2, lacking the label of *node 5* will make *node 1* and *node 4* be incorrectly recognized as homophilous nodes. The second challenge is that the GNN model is hard to be trained to learn nodes of homophily and heterophily adaptively. For homophilous nodes, messages passing from neighbors can be useful since they have the same label. But heterophilous nodes may not extract beneficial information from neighbors due to the diverse classes they belong to. Therefore, it is challenging to train an unbiased GNN on semi-homophilous graphs with a balanced performance on both homophilous and heterophilous nodes.

To address the above challenges, we propose an Uncertainty-aware Debiasing (UD) framework to mitigate the bias issue on semi-homophilous graphs. For the first challenge, we differentiate heterophilous nodes from homophilous ones by using uncertainty of GNN predictions instead of node labels. The idea is that GNN exhibits high model uncertainty on heterophilous nodes where the GNN aggregates diverse messages from neighbors. As such, the predictions of heterophilous nodes will show a wider variance. In this light, we devise an uncertainty estimation module to approximate the model uncertainty of GNN predictions. For the second challenge, we propose a debiased training module to obtain a debiased prediction based on the estimated uncertainty and the output of the biased model. In particular, we prune the biased GNN model by removing some redundant weights and retraining it with certain nodes. Then the redundant weights are trained from scratch on nodes with high uncertainty scores to mitigate the bias.

We integrate the above two stages of uncertainty estimation and debiased training into general GNN frameworks and name our model as Uncertainty-aware Debiased Graph Neural Network (UD-GNN). Our contributions can be listed as follows.

- We reveal a severe bias issue of current GNNs regarding node-level homophily and propose an uncertainty-aware debiasing framework to obtain debiased GNNs.
- We design two universal modules for uncertainty estimation and debiased training, which estimates the uncertainty of GNN predictions and mitigates the bias towards the homophilous nodes, respectively.
- Extensive experiments are conducted on both synthetic and benchmark datasets to verify the rationality and effectiveness of the proposed framework.

2 PRELIMINARIES

In this work, we denote a graph as $\mathcal{G} = (\mathcal{V}, \mathbf{A}, \mathbf{X})$, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ is the set of nodes, $\mathbf{X} \in \mathbb{R}^{N \times D}$ denotes node features, and $\mathbf{A} \in \{0,1\}^{N \times N}$ is an adjacency matrix representing the connections between nodes. Note that N and D represent the number of nodes and features, respectively. Besides, we denote the labels of node as $\mathbf{Y} \in \{0,1\}^{N \times C}$ where C is the number of classes.

2.1 Homophily Measure

Definition 2.1 (Graph-level Homophily Ratio). Given a graph $\mathcal{G} = (\mathcal{V}, \mathbf{A}, \mathbf{X})$ and the node labels $\mathbf{Y}$, graph-level homophily ratio $h_{\mathcal{G}}$ is defined as the fraction of intra-class edges. Formally, $h_{\mathcal{G}} = \frac{|\mathcal{E}_{\text{intra}}|}{|\mathbf{A}|}$, where $\mathcal{E}_{\text{intra}} = \{(u,v) \mid A_{uv} = 1 \wedge \mathbf{y}_u = \mathbf{y}_v\}$.

Definition 2.2 (Node-level Homophily Ratio). As to a node v in $\mathcal{G}$, the node-level homophily ratio h_v is defined as $h_v = \frac{|\mathcal{E}_v \cap \mathcal{E}_{\text{intra}}|}{|\mathcal{E}_v|}$, where $\mathcal{E}_v = \{(u,v) \mid A_{uv} = 1\}$ is the set of edges linked to v.

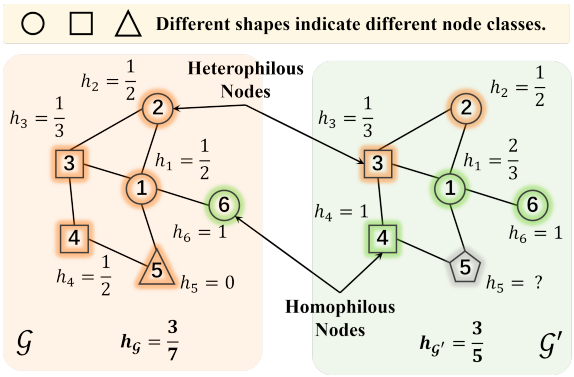


Figure 2: Demonstration of graph-level and node-level homophily ratio. If the label of node 5 is unknown, two heterophilous nodes (node 1 and 4 in the Left) change to be homophilous (Right) from the perspective of node-level homophily ratio. The Right graph-level homophily ratio is higher than the Left. The partially-available information gives unreliable homophily measures.

According to the node-level homophily ratio, $\mathcal{V}$ can be divided into homophilous node set $\mathcal{V}_{\text{homo}}$ and heterophilous node set $\mathcal{V}_{\text{heter}}$, satisfying $\mathcal{V}_{\text{homo}} \cup \mathcal{V}_{\text{heter}} = \mathcal{V}$ and $\mathcal{V}_{\text{homo}} \cap \mathcal{V}_{\text{heter}} = \emptyset$.

An example of graph-level and node-level homophily ratio is demonstrated in Figure 2. In graph $\mathcal{G}$, 3 of 7 edges connect two nodes with the same label thus the graph-level homophily ratio is $\frac{3}{7}$. If one of the node label is unknown, as shown in graph $\mathcal{G}'$, edges with unknown node labels are not taken into account.

2.2 Categorization of Graph Benchmarks

We systematically categorize graph benchmarks from two-level homophily measure. **Homophilous graphs** have relatively high graph-level homophily ratio close to 1, where Cora, Citeseer, and PubMed are typical examples. **Non-homophilous graphs** have relatively low graph-level homophily ratio near 0 and are summarized in [25] and [20].

Non-homophilous graphs can be further categorized to **semi-homophilous graphs** and **heterophilous graphs** from the perspective of node-level homophily ratio. As illustrated in Figure 1, heterophilous nodes are grouped into strong heterophily or weak heterophily according to the level of homophily ratio and so are homophilous nodes. **Semi-homophilous graphs** have nodes evenly distributed across four groups and none of them exceeds half of the whole. While **heterophilous graphs** have over 50% nodes belonging to strong heterophily.

2.3 Problem Statement

We mainly follow the common setting of node classification, where the target is to learn a C-way classification function to predict the unlabeled nodes in a semi-homophilous graph. Formally,

$$f_{\mathbf{W}} : (\mathbf{A}, \mathbf{X}) \rightarrow \mathbf{Y}, \tag{1}$$

where $\mathbf{W}$ denotes the parameters of the function. The parameters are typically learned over a set of labeled nodes by minimizing a classification loss such as cross-entropy.

The existing studies design different GNNs as the classification function, which are largely focused on the overall classification accuracy. Our setting further emphasizes the bias issue of GNNs and requires the model prediction to maintain both high accuracy and *low model bias*. Here the model bias refers to that GNNs trained on semi-homophilous graphs perform worse on heterophilous set than the same model trained only on the heterophilous part of the graph, resulting in a larger performance gap between homophilous and heterophilous nodes.

Relation to heterophily. The benchmark graphs are different. Our main focus is to mitigate the bias of GNNs on semi-homophilous graphs while GNNs designed for heterophily aim to improve the performance on heterophilous graphs.

Relation to fairness. The assumptions are different. The group disparity-based fairness model assumes that the outcome is invariant to the group but this does not hold in our case. The disparity between homophily and heterophily is not necessarily zero because the baselines for them are different.

3 METHODOLOGY

In this section, we present the proposed UD-GNN framework. Firstly, we give an overview of the whole framework. Then, we detail the uncertainty estimation process in Section 3.2 and debiased training process in Section 3.3.

3.1 Overview

We illustrate the pipeline of the proposed framework on an example graph in Figure 3, which consists of two modules: uncertainty estimation and debiased training.

In the uncertainty estimation module, we adopt Monte Carlo dropout variational inference [8] method to estimate the model uncertainty of the prediction. The training node set is divided into a confident node set and an uncertain node set according to their uncertainty scores. In the debiased training module, the biased weights are pruned to remove redundant parameters and retrained with the confident set. The pruned parameters are trained from scratch with the uncertain set to mitigate the bias issue.

3.2 Uncertainty Estimation

Due to the difficulty in extracting information from heterophilous neighborhood, the output of GNNs may exhibit relatively high model uncertainty for heterophilous nodes. To capture model uncertainty, Bayesian graph neural networks replace the biased GNN's weight parameters $\mathbf{W}_b$ with distributions over these parameters, like a Gaussian prior distribution $\mathbf{W} \sim \mathcal{N}(0, I)$. Given the graph data $(\mathbf{A}, \mathbf{X})$ and the prediction $\hat{\mathbf{Y}}$, the model likelihood for node classification can be defined as Eq. (2).

$$p(\hat{\mathbf{Y}} \mid \mathbf{A}, \mathbf{X}) = \int_{\mathbf{W}} p(\hat{\mathbf{Y}} \mid \mathbf{W}, \mathbf{A}, \mathbf{X}) p(\mathbf{W} \mid \mathbf{A}, \mathbf{X}) \mathrm{d}\mathbf{W} \tag{2}$$

To overcome the intractable inference of evaluating the true model posterior $p(\mathbf{W}|\mathbf{A}, \mathbf{X})$ in Eq. (2), we utilize Monte Carlo dropout variational inference [8] to approximate it, which defines a simple distribution $q_\theta(\mathbf{W})$ by multiplying a deterministic model weight $\mathbf{W}_b$ with a binary mask M following Bernoulli distribution with parameter θ, as demonstrated in Eq. (3).

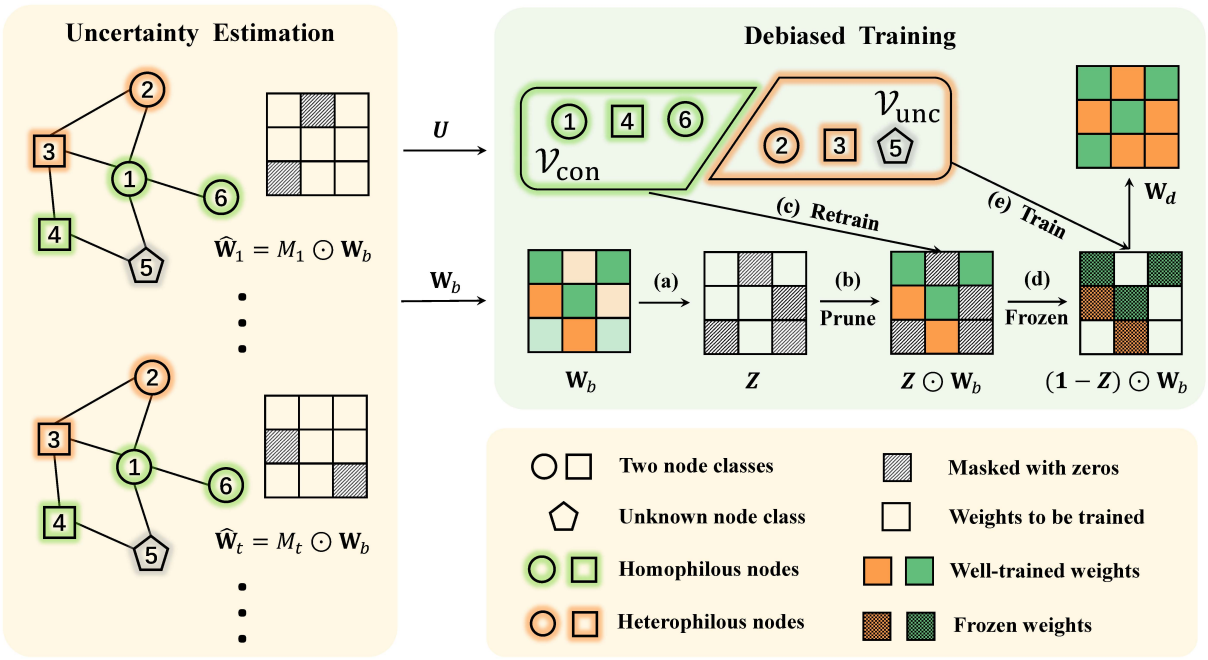


Figure 3: The framework of UD-GNN on an example graph. The input is a graph with three homophilous nodes (1,4,6), two heterophilous nodes (2,3) and one node (5) without label. In the uncertainty estimation module, $\widehat{\mathbf{W}}_1, \ldots, \widehat{\mathbf{W}}_T$ are sampled from the dropout distribution, and the nodes are divided into confident set $\mathcal{V}_{\text{con}}$ and uncertain set $\mathcal{V}_{\text{unc}}$ according to the uncertainty score. The obtained parameter $\mathbf{W}_b$ is biased. In the debiased training module, the redundant weights in $\mathbf{W}_b$ are identified with Z (a). Then $\mathbf{W}_b$ is pruned (b) and retrained with $\mathcal{V}_{\text{con}}$ (c). With these parameters frozen (d), the other parameters are trained with $\mathcal{V}_{\text{unc}}$ (e) and the resulting $\mathbf{W}_d$ is debiased.

$$\begin{aligned} P(M) &\sim \text{Bernoulli}(\theta) \\ q_\theta(\mathbf{W}) &\sim M \odot \mathbf{W}_b \end{aligned} \tag{3}$$

During the MC dropout variational inference, $\{\hat{\mathbf{W}}_t\}_{t=1}^T$ are sampled from $q_\theta(\mathbf{W})$. $\hat{Y}_t = f_{\hat{\mathbf{W}}_t}(\mathbf{A}, \mathbf{X})$ is the prediction under the sampled weight $\hat{\mathbf{W}}_t$. The deterministic model weight $\mathbf{W}_b$ is trained by minimizing the cross-entropy loss defined in Eq. (4), where θ is a hyper-parameter.

$$\mathcal{L}(\mathbf{W}_b) = -\frac{1}{T}\sum_{t=1}^{T} \mathbf{Y} \log\left(f_{\hat{\mathbf{W}}_t}(\mathbf{A}, \mathbf{X})\right) + \frac{1-\theta}{2T}\|\mathbf{W}_b\|^2 \tag{4}$$

After the training process, the model uncertainty score U is estimated by Eq. (5), which is a N-dimensional vector indicating the uncertainty score of each node.

$$\begin{aligned} \mathbb{E}[\hat{\mathbf{Y}} \mid \mathbf{A}, \mathbf{X}] &= \int \hat{\mathbf{Y}} p(\hat{\mathbf{Y}} \mid \mathbf{A}, \mathbf{X}) \mathrm{d}\hat{\mathbf{Y}} \approx \frac{1}{T}\sum_{t=1}^{T} \hat{Y}_t \\ U(\hat{\mathbf{Y}} \mid \mathbf{A}, \mathbf{X}) &= \text{Var}(\hat{\mathbf{Y}} \mid \mathbf{A}, \mathbf{X}) \approx \frac{1}{T}\sum_{t=1}^{T}\left(\hat{Y}_t - \mathbb{E}[\hat{\mathbf{Y}} \mid \mathbf{A}, \mathbf{X}]\right)^2 \end{aligned} \tag{5}$$

3.3 Debiased Training

The biased parameters are denoted by $\mathbf{W}_b$, and the well-trained graph neural network is represented by $f_{\mathbf{W}_b}$. The purpose of this subsection is to train a debiased graph neural network $f_{\mathbf{W}_d}$ from $f_{\mathbf{W}_b}$, with the performance gap to be mitigated.

The basic idea is to restrict the complexity of the biased model on the confident nodes and restore the model capacity for the uncertain nodes. Inspired by continual learning [9, 17], we prune the model parameters for the confident nodes by replacing part of parameters with zero and freeze the other ones. Then parameters which have been set to zero are retrained with the uncertain nodes.

Specifically, the training nodes, denoted by $\mathcal{V}_{\text{train}}$, can be divided into a confident node set $\mathcal{V}_{\text{con}}$ and an uncertain node set $\mathcal{V}_{\text{unc}}$ according to the uncertainty score $U(\hat{\mathbf{Y}} \mid \mathbf{A}, \mathbf{X}_{\text{train}})$ estimated by Eq. (5). The debiasing ratio γ is defined to be $\gamma = \frac{|\mathcal{V}_{\text{con}}|}{|\mathcal{V}_{\text{unc}}|}$, where the two sets satisfy $\mathcal{V}_{\text{con}} \cup \mathcal{V}_{\text{unc}} = \mathcal{V}_{\text{train}}$ and $\mathcal{V}_{\text{con}} \cap \mathcal{V}_{\text{unc}} = \emptyset$.

The confident node set $\mathcal{V}_{\text{con}}$ refers to nodes with low uncertainty score. Due to the bias of $\mathbf{W}_b$, $f_{\mathbf{W}_b}$ has learnt a good representations for $\mathcal{V}_{\text{con}}$. The graph neural network $f_{\mathbf{W}_b}$ can be pruned by removing unimportant parameters in $\mathbf{W}_b$ of which the absolute value is almost zero[3]. Z is a $0-1$ pruning mask of $\mathbf{W}_b$ with 0 in the position of pruned parameters and 1 otherwise.

After pruning, the GNN performance can be affected because of the big changes in the network structure. We perform retraining over $Z \odot \mathbf{W}_b$ on $\mathcal{V}_{\text{con}}$ so as to regain the knowledge learnt by the biased model. After this step, there are some free parameters $(1 - Z) \odot \mathbf{W}_b$, which will be optimized on $\mathcal{V}_{\text{unc}}$.

[3]We simply set the threshold to be 1e-5 and weights smaller than that would be pruned.

Algorithm 1: UD-GNN: Uncertainty-aware Debiased Graph Neural Network

Input: $\mathcal{G} = (\mathcal{V}, \mathbf{A}, \mathbf{X})$: A semi-homophilous graph, $\mathcal{V}_{\text{train}}$: Set of training nodes, $\mathbf{Y}$: Label of training nodes, T: Number of Monte Carlo samples, θ: Dropout rate for uncertainty estimation, γ: Debiasing ratio.

Output: The debiased parameters $\mathbf{W}_d$.

1 Choose message passing architectures for biased GNN $f_{\mathbf{W}_b}$ and debiased GNN $f_{\mathbf{W}_d}$;
2 Initialize parameters $\mathbf{W}_b$ and $\mathbf{W}_d$;
3 **for** $t = 1, \ldots, T$ **do**
4 Sample a weight distribution $\hat{\mathbf{W}}_t$ from $q_\theta(\mathbf{W})$ w.r.t. (3);
5 Train $\mathbf{W}_b$ and $\{\hat{\mathbf{W}}_t\}_{t=1}^T$ by minimizing Eq. (4);
6 Estimate the model uncertainty U w.r.t. Eq. (5);
7 Divide $\mathcal{V}_{\text{train}}$ into $\mathcal{V}_{\text{con}}$ and $\mathcal{V}_{\text{unc}}$ according to U such that $\gamma = \frac{|\mathcal{V}_{\text{con}}|}{|\mathcal{V}_{\text{unc}}|}$;
8 Prune the network $Z \odot \mathbf{W}_b$ and retrain with $\mathcal{V}_{\text{con}}$;
9 Freeze $Z \odot \mathbf{W}_b$ and train $(1 - Z) \odot \mathbf{W}_b$ in $\mathbf{W}_d$ by minimizing Eq. (7);
10 return $\mathbf{W}_d$.

The uncertain node set $\mathcal{V}_{\text{unc}}$ contains nodes with high uncertainty, which indicates that the model is not sure about the predictions of these nodes. $\mathbf{W}_d$ is partially trained to minimize the cross-entropy loss Eq. (7) on $\mathcal{V}_{\text{unc}}$, with the retrained parameters $Z \odot \mathbf{W}_b$ frozen as shown in Eq. (6).

$$\mathbf{W}_d = (1 - Z) \odot \mathbf{W}_b + \text{stop\_gradient}(Z \odot \mathbf{W}_b) \tag{6}$$

$$\mathcal{L}(\mathbf{W}_d) = -\frac{1}{|\mathcal{V}_{\text{unc}}|} \sum_{v \in \mathcal{V}_{\text{unc}}} y_v \log\left(f_{\mathbf{W}_d}(\mathbf{A}, \mathbf{X}_{v \cup \mathcal{N}_v})\right) \tag{7}$$

The overall training algorithm is summarized in Algorithm 1. Given a semi-homophilous graph $\mathcal{G}$ and the training node set $\mathcal{V}_{\text{train}}$, we first choose the GNN architecture for $f_{\mathbf{W}_b}$ and $f_{\mathbf{W}_d}$ (Line 1) then initialize $\mathbf{W}_b$ and $\mathbf{W}_d$ (Line 2). Next, we sample T weight distributions for dropout variational inference (Line 4). The biased weight together with sampled weights is trained to minimize Eq. (4) (Line 5). After that, the uncertainty score of each node can be estimated (Line 6), and $\mathcal{V}_{\text{train}}$ is divided into $\mathcal{V}_{\text{con}}$ and $\mathcal{V}_{\text{unc}}$ (Line 7). The network is pruned with $Z \odot \mathbf{W}_b$ and retrained with $\mathcal{V}_{\text{con}}$ (Line 8). Finally, the masked parameters are frozen and the unmasked parameters are optimized by minimizing Eq. (7) (Line 9).

In the testing phase, the testing nodes, denoted by $\mathcal{V}_{\text{test}}$, can be divided into the confident node set and the uncertain node set according to the debiasing ratio γ in the training set. For the confident testing nodes, the predictions are obtained from $f_{Z \odot \mathbf{W}_d}$ and other predictions are conducted by $f_{\mathbf{W}_d}$.

4 EXPERIMENTS

In this section, we investigate the effectiveness of the proposed UD-GNN model on both synthetic and benchmark datasets, which belong to semi-homophilous graphs, with the aim of answering the following research questions.

- **RQ1**: Does UD-GNN outperform the state-of-the-art methods on semi-homophilous graphs?
- **RQ2**: How do the key components contribute to the results?
- **RQ3**: Why does uncertainty estimation benefit the debiasing between homophily and heterophily?
- **RQ4**: Does UD-GNN work well on heterophilous graphs?
- **RQ5**: What is the sensitivity of UD-GNN with respect to different debiasing ratios, mixing ratios and number of classes?

4.1 Experimental Setup

4.1.1 Datasets. As depicted in Figure 1, we conduct experiments on three semi-homophilous graphs, collected from non-homophilous benchmarks [20]. High-homophilous graphs are not our focus since they do not suffer from the bias issue. Besides, we construct a synthetic semi-homophilous graph based on cSBM [4]. The mixing ratio in cSBM is defined as the fraction of homophilous nodes and heterophilous nodes, which is further analysed in Section 4.6. The statistics of the datasets are reported in Table 1. Nodes with node-level homophily ratios smaller than 0.5 are regarded as heterophilous nodes, and we report the fraction of heterophilous nodes in each dataset as shown in the table. Detailed descriptions can be referred to Appendix A.1.

Table 1: Statistics of the datasets.

Dataset	#Node	#Edge	#Class	#Feat	%Heter
cSBM	20,000	998,766	10	1,024	50%
Penn94	41,554	1,362,229	2	4,814	49%
Cora-full	19,793	126,842	70	8,710	44%
Ogbn-arxiv	169,343	1,166,243	40	128	37%

4.1.2 Compared Methods. We compare with several state-of-the-art graph neural network methods, including GCN [19], GAT [33], Mixhop [1], GPR-GNN [3], JK-Net [36], H2GCN [41], CPGNN [40], WRGAT [31], U-GCN [16], to verify the effectiveness of our proposed method on semi-homophilous graphs. Details of the compared methods can be found in Appendix A.2.

UD-GNN is a general name for all GNNs equipped with our Uncertainty-aware Debiasing (UD) framework. For GCN, GAT, Mixhop, GPR-GNN, VA denotes the vanilla version and UD is the proposed framework. We also derive two variants of UD-GNN to compare and analyze the performances of its each component. **D-GNN** removes uncertainty estimation and trains a separate model to discriminate homophilous nodes from heterophilous nodes for debiasing. **U-GNN** removes debiased training and applies Focal loss based on the estimated uncertainty scores.

4.1.3 Experimental Settings. For all the compared methods, we report the average value and standard deviation of 5 runs.[4] The hyper-parameters in UD-GNN include the number of samples T, dropout rate θ, and the debiasing ratio γ. T is used for uncertainty estimation and is set to be 5 in our experiments. Since the base GNN is efficient and fast, running T times of the model would

[4]Exceptionally, the experiment of H2GCN on ogbn-arxiv was conducted on CPU due to the out-of-memory issue in GPU. It takes around a week for a run thus we only report the results of this run and the deviation is 0.

Table 2: Performance comparison for node classification on semi-homophilous graphs. The best results are bolded.

Dataset		cSBM			Penn94			Cora-full			Ogbn-arxiv		
Metric		Acc ↑	δ_{ho} ↓	δ_{he} ↓	Acc ↑	δ_{ho} ↓	δ_{he} ↓	Acc ↑	δ_{ho} ↓	δ_{he} ↓	Acc ↑	δ_{ho} ↓	δ_{he} ↓
GCN	VA	47.30±0.43	1.23	9.11	82.06±0.19	3.89	3.32	68.81±0.29	2.8	3.11	71.17±0.11	**0.03**	**7.76**
	UD	49.39±0.16	2.31	4.42	82.96±0.21	3.93	1.31	69.70±0.24	3.51	1.12	72.06±0.17	**1.88**	**0.28**
GAT	VA	50.44±0.29	1.68	8.27	79.85±0.73	5.48	8.63	69.24±0.27	1.82	3.27	69.90±0.12	0.43	6.24
	UD	52.66±0.31	2.04	3.12	83.54±0.20	5.24	0.89	70.85±0.34	2.21	1.18	71.07±0.35	1.65	0.65
Mixhop	VA	51.31±0.41	0.93	8.25	82.14±0.21	2.37	5.36	69.11±0.31	2.26	3.94	72.80±0.23	0.7	5.63
	UD	52.51±0.17	1.84	3.17	**83.68±0.65**	3.22	2.01	70.01±0.34	2.94	2.67	**73.20±0.09**	2.13	0.2
GPRGNN	VA	53.79±0.15	**1.29**	**9.23**	76.77±0.25	**5.58**	**10.92**	70.15±0.30	**1.61**	**4.92**	70.98±0.17	0.95	5.84
	UD	**54.80±0.20**	**1.62**	**2.48**	80.70±0.24	**6.82**	**0.91**	**71.09±0.30**	**1.65**	**2.36**	71.32±0.13	2.48	1.04
JK-NET		50.68±0.38	2.67	6.21	81.26±0.23	3.35	8.24	68.12±0.23	2.34	3.23	70.66±0.19	0.72	8.23
H2GCN		51.93±0.25	2.46	8.26	81.63±0.16	4.16	7.12	70.82±0.82	3.35	3.67	70.14±0.00	0.28	6.82
CPGNN		51.84±0.67	2.03	5.26	80.92±0.67	4.12	8.21	70.38±0.23	2.35	3.26	69.24±0.52	0.52	6.27
WRGAT		52.48±0.28	1.89	6.26	82.32±0.83	3.12	6.23	71.32±0.92	2.92	4.38	71.23±0.67	0.46	6.23
U-GCN		52.74±0.72	1.73	8.23	82.31±0.89	5.23	4.21	70.47±0.78	1.92	4.21	70.27±0.81	0.61	5.98

not impose much computational cost. γ depends on the fraction of heterophilous nodes in the dataset and is tuned around $1:1$ in most cases. Details can be referred to Appendix A.4.

4.1.4 Metrics. We adopt **Accuracy** to measure the performance of all the compared methods. The difference of Accuracy between the ideal model and the actual model is used to evaluate the relative model bias, denoted by $\delta = \text{eval}(\mathbf{Y}, f_{\mathbf{W}^*}(\mathbf{A}, \mathbf{X})) - \text{eval}(\mathbf{Y}, f_{\mathbf{W}}(\mathbf{A}, \mathbf{X}))$. Here, eval refers to the evaluation function like Accuracy and $\mathbf{W}^*$ denotes the intended model parameters without bias. We compute δ on the homophilous set and the heterophilous set, denoted by δ_{ho} and δ_{he} respectively. Higher accuracy indicates better performance of an approach, and lower δ represents lower model bias.

The intended parameters are approximated by training GNNs only on the homophilous or heterophilous set to avoid possible model bias. The highest result of 5 runs is chosen to be the intended score for evaluation and the result relies on specific model architectures.

4.2 Performance Comparison (RQ1)

To answer RQ1, we evaluate the performance of all the compared GNNs on four semi-homophilous graph datasets. The scores of accuracy and relative bias are reported in Table 2. We have the following observations.

Firstly, the relative bias for both homophilous and heterophilous nodes exists for all compared methods, with heterophilous bias δ_{he} larger than homophilous bias δ_{ho}. We conclude that graph neural networks always suffer from the bias issue and the classification accuracy would be improved if such bias can be mitigated.

Secondly, combining the UD framework with GCN, GAT, Mixhop, and GPR-GNN consistently mitigates the bias issue since the accuracy of UD is higher than VA and the heterophilous bias δ_{he} of UD is much lower than VA. The homophilous bias δ_{ho} of UD increases a little compared with VA because the original network for the confident set is pruned. However, the overall bias score of UD, i.e. the sum of δ_{ho} and δ_{he}, declines consistently and the best results in Table 2 are bolded in terms of the sum bias. Based on the results of uncertainty estimation, the debiased training strategy is effective in debiasing GNNs between homophily and heterophily.

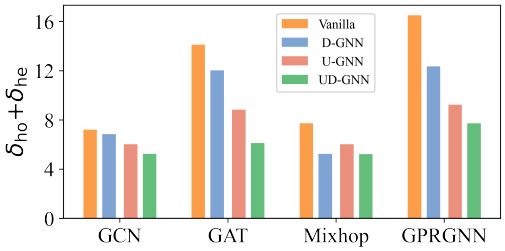
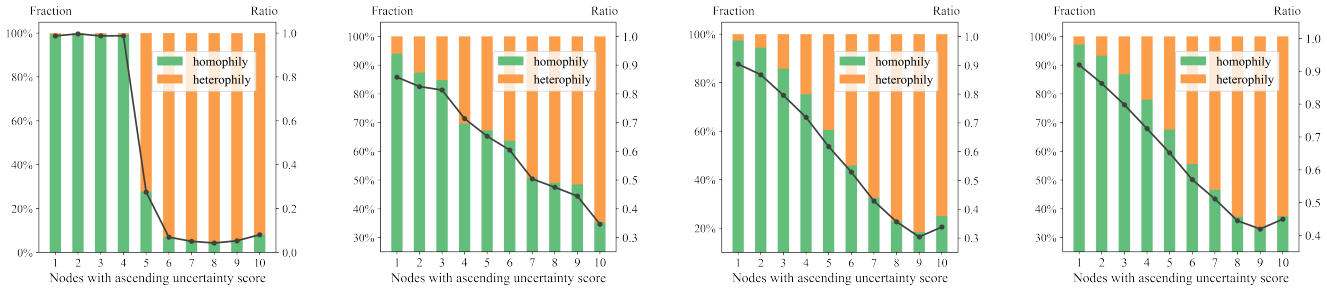


Figure 4: Ablation Study of UD-GNN on Penn94.

Finally, the combination of UD framework with GNNs exhibits different performances for different datasets. For cSBM and Cora-full, UD-GPR-GNN achieves the best accuracy and the lowest bias δ. For Penn94 and Ogbn-arxiv, the accuracy of UD-Mixhop is the highest while UD-GPRGNN and UD-GCN achieve the most bias reduction respectively. We think this is due to the complexity of the datasets. For benchmark datasets Penn94 and Ogbn-arxiv, heterophilous nodes may imply complex patterns, thus Mixhop with high-order neighbors can capture these regularities more accurately. The performance of heterophilous nodes is higher, and the gap is smaller. For synthetic dataset cSBM, since the connection rule is artificially designed, GNNs using one-hop neighbors can work well.

4.3 Ablation Study (RQ2)

To answer RQ2, we identify two key modules of UD-GNN, namely uncertainty estimation and debiased training, and verify their effectiveness by removing each module respectively. Due to the limited space, we only report the results of Penn94 in Figure 4 for demonstration and the other three datasets also exhibit similar trends.



(a) cSBM (b) Penn94 (c) Cora-full (d) Ogbn-arxiv

Figure 5: The relationship between estimated uncertainty score and node-level homophily ratio. For the four datasets, nodes are equally divided into 10 bins according to the sorted uncertainty score. The x-axis denotes 10 bins of nodes with ascending uncertainty scores. The left y-axis indicates the fraction of homophilous and heterophilous nodes in each bin. The right y-axis corresponds to the average node-level homophily ratio in each bin, which is shown by the broken line.

Since our main focus is the bias issue, we only report the relative bias and denote the sum of δ_{ho} and δ_{he} as δ below for brevity. The full model UD-GNN always achieves the lowest δ compared with the two variants D-GNN and U-GNN, which indicates that each module is necessary for mitigating the bias issue.

In addition to the vanilla GNN, D-GNN achieves the second highest bias value. It is hard to train a classifier to identify homophilous or heterophilous nodes since partial label information gives unreliable homophily measures, as mentioned in Figure 1. The bias of D-GNN is large due to the estimation error of the trained homophily classifier in the variant.

For U-GNN, the score of δ is always higher than UD-GNN. The focal loss can reweight uncertain nodes and confident nodes in the back-propagation step from the loss function to prevent the model from getting biased to homophilous nodes but the effectiveness is weaker than the debiased training which UD-GNN does.

4.4 Extensive Study (RQ3)

To answer RQ3, we demonstrate the output of the uncertainty estimation module from GAT in Figure 5. For the four datasets, we divide nodes into 10 bins equally according to the sorted uncertainty score. The left y-axis indicates the fraction of homophilous and heterophilous nodes in each bin. The right y-axis corresponds to the average node-level homophily ratio in each bin. As the uncertainty increases, the fraction of heterophily gets larger, and the average node-level homophily ratio gets lower. Especially, the node-level homophily ratio in cSBM is either 0 (strong heterophily) or 1 (strong homophily). Therefore, the average ratio for each bin lies exactly in the boundary of two bars. It is concluded that the uncertainty estimation module is effective in differentiating heterophily from homophily, and the resulting debiased training module gets meaningful division of nodes for further training.

4.5 Heterophilous Graphs Performance (RQ4)

To answer RQ4, we further evaluate the performance of UD-GNN on three heterophilous graphs, namely Chameleon, Squirrel, Wisconsin. The detailed description can be found in Appendix A.1 and the results are reported in Table 3. Since most nodes in the datasets are heterophilous, we only test on heterophilous GNNs and do not compute model bias δ. On heterophilous benchmarks, UD-GNN improves the performance due to the refining of uncertain nodes in the debiased training and achieves the best results on Chameleon and Squirrel, with comparable performance on Wisconsin.

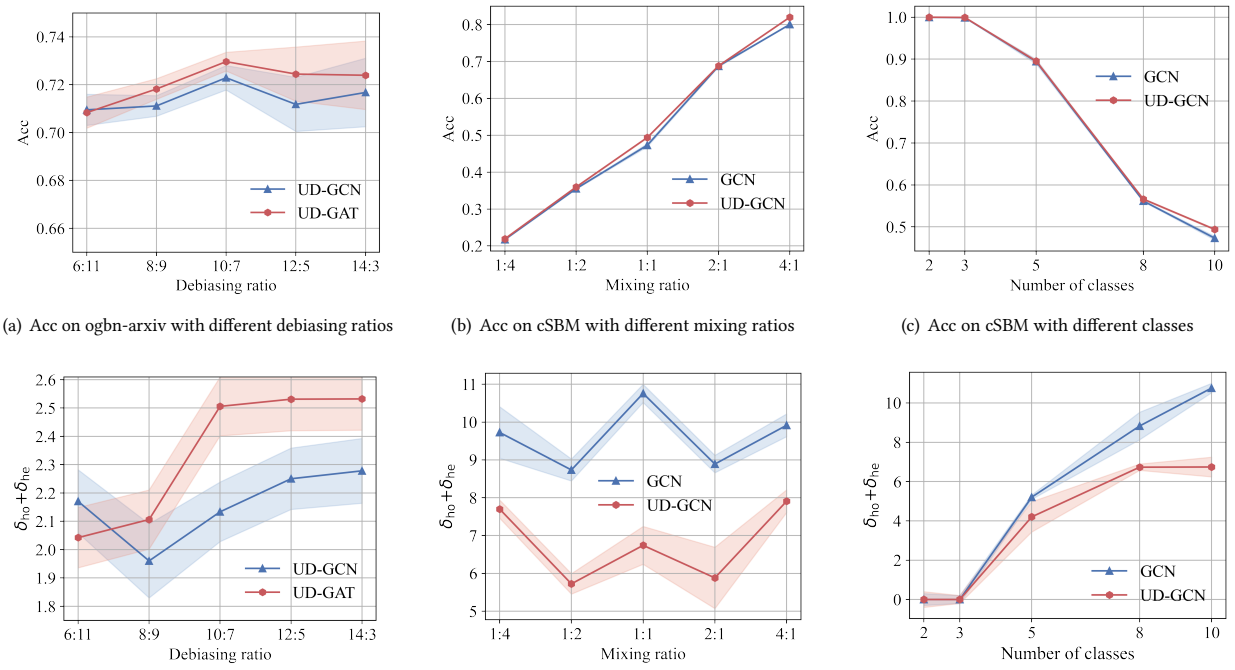
Table 3: Performance on Heterophilous Graphs.

Dataset		Chameleon	Squirrel	Wisconsin
Mixhop	VA	58.25±1.83	42.86±1.48	73.83±6.82
	UD	59.23±1.24	43.92±1.42	74.23±5.92
GPRGNN	VA	64.36±0.87	46.83±0.84	79.23±3.81
	UD	66.23±1.03	47.92±0.25	81.29±2.34
JK-NET		53.95±1.14	33.51±1.32	48.39±5.28
H2GCN		57.39±1.96	35.23±1.35	85.88±4.92
CPGNN		59.11±1.23	36.27±1.29	86.29±4.28
WRGAT		63.26±1.67	41.26±1.37	86.28±2.46
U-GCN		54.07±1.57	34.39±1.34	69.89±2.54

4.6 Sensitivity Analysis (RQ5)

To answer RQ5, we further evaluate the sensitivity of UD-GNN from three aspects, namely debiasing ratio, mixing ratio, and the number of classes. Due to the limited space, we only report the results of UD-GCN or UD-GAT for better visualization. Other UD-GNNs also exhibit similar trends of sensitivity. The sum of δ_{ho} and δ_{he} is denoted as δ below for brevity.

The debiasing ratio is the threshold that divides the nodes into the confident set and the uncertain set, which has been defined in Section 3.3. We evaluate the sensitivity of UD-GCN and UD-GAT on ogbn-arxiv, and the results are shown in Figure 6(a) and Figure 6(d). With the size of confident nodes increasing, the accuracy rises to the top then falls, and the δ increases then keeps stable. We conclude that there is a trade-off between pruning and debiasing for UD framework. If the size of uncertain nodes is larger, the performance of homophily would decline. And if the size of confident nodes is larger, the model would be easily biased.



(a) Acc on ogbn-arxiv with different debiasing ratios

(b) Acc on cSBM with different mixing ratios

(c) Acc on cSBM with different classes

(d) δ on ogbn-arxiv with different debiasing ratios

(e) δ on cSBM with different mixing ratios

(f) δ on cSBM with different classes

Figure 6: Sensitivity of UD-GNN with respect to debiasing ratio (a,d), mixing ratio (b,e), and number of classes (c,f). The marker point represents the mean value of 5 runs, and the shaded area corresponds to the standard deviation.

The mixing ratio measures the extent of mixing between homophilous and heterophilous nodes. We adjust the mixing ratio from 1 : 4 to 4 : 1 and test the performance of GCN and UD-GCN. As demonstrated in Figure 6(b) and Figure 6(e), with the fraction of heterophily decreasing, the overall accuracy increases due to the increase of homophily. Our UD-GCN framework consistently mitigates the bias effectively under different settings.

Finally, we change the number of classes in the synthetic dataset to test the bias of GCN and debiasing ability of UD-GCN. As shown in Figure 6(c) and Figure 6(f), when the number of classes is smaller than 3, there is no bias since the accuracy is around 1 and the δ is near 0. As the number of classes grows larger than 5, the accuracy declines, and the δ increases, which indicates the bias issue. UD-GCN is effective in debiasing GCN since it achieves lower bias and maintains higher accuracy.

5 RELATED WORK

This section introduces previous studies on heterophily and uncertainty in graph neural networks.

5.1 Heterophily in Graph Neural Networks

Heterophily [26] was first proposed as a concept in communication research, which refers to individuals who interact are different with respect to certain attributes. Opposite to this concept, homophily has been widely investigated in social networks [23]. Graph neural networks have been shown as low-pass filters [24] and can be characterized by smoothing and de-noising node features [7], thus tend to work well on homophily and fail on heterophily.

Several kinds of strategies can be applied to tackle graphs with heterophily. A direct way is to model the label similarity of the connected nodes, like GAM [30] and CS-GNN [10]. Another way is to design trainable weights for the aggregation step in GNNs. GPR-GNN [3] combines Generalized PageRank scheme with GNNs that learns the GPR weights adaptively. LA-GCN [37] unifies a learnable aggregator for GCN and assigns different importances to both nodes and features.

Modeling label similarities or designing trainable weights can model the heterophily explicitly or implicitly. However, they rely on the node labels for training. Different from these two strategies, the proposed uncertain estimation can discriminate heterophily from homophily without the help of node labels.

Besides, adding other information like higher-order neighbors or label compatibility has been demonstrated to be helpful on graphs with heterophily. H2GCN [41] identifies three key designs for boosting heterophily learning. CPGNN [40] incorporates an interpretable compatibility matrix for modeling the heterophily level in the graph, which can be learned in an end-to-end fashion. WRGAT [31] improves the assortativity of graphs by constructing a computational graph based on structural information. U-GCN [16] is a new universal GCN framework extracting information from 1-hop, 2-hop and kNN networks simultaneously.

The methods mentioned above remedy the limitations of current GNNs on heterophilous graphs. Different from them, our major concern in this work is the bias issue of GNNs trained on semi-homophilous graphs.

5.2 Uncertainty in Graph Neural Networks

Predictive uncertainty estimation [18] has been explored in deep neural networks with Bayesian approximation. For graph neural networks, multidimensional uncertainty types can also be estimated. S-BGNN [38] is a multi-source uncertainty framework of GNNs for misclassification detection and out-of-distribution detection. EV-GCN [15] designs a learnable adaptive population graph and achieves superior performance on brain analysis and ocular disease prediction. UAG [5] incorporates a Bayesian uncertainty technique to develop an uncertainty-aware attention for adversarial attacks on GNNs. CGI [6] estimates the causal effect of a node's local structure for the prediction using Monte Carlo estimation.

The above methods investigate uncertainty in GNNs for different tasks like disease prediction, adversarial attack, misclassification detection, etc. Different from them, we estimate the uncertainty in GNNs to mitigate the bias issue for semi-homophilous graphs and propose an uncertainty-aware debiasing framework to narrow the performance gap between homophilous and heterophilous nodes.

6 CONCLUSION

In this work, we investigate a challenging task for graph neural networks on semi-homophilous graphs, i.e. the bias issue between homophily and heterophily. Suffering from this issue, graph neural networks always exhibit a performance gap between homophilous and heterophilous nodes. To overcome this problem, we propose an Uncertainty-aware Debiasing framework which retrains the parameters on nodes with high uncertainty, and inherits the outputs of the biased model for nodes with low uncertainty. Through our extensive experiments on both synthetic and benchmark datasets, UD-GNN is verified to effectively debias graph neural networks on semi-homophilous graphs.

ACKNOWLEDGMENTS

The research work is supported by the National Natural Science Foundation of China under Grant (No.61976204, U1811461, U1836206). Xiang Ao is also supported by the Project of Youth Innovation Promotion Association CAS, Beijing Nova Program Z201100006820062. Yang Liu is also supported by China Scholarship Council.

REFERENCES

[1] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, et al. 2019. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *ICML*. PMLR, 21–29.
[2] Aleksandar Bojchevski and Stephan Günnemann. 2018. Deep Gaussian Embedding of Graphs: Unsupervised Inductive Learning via Ranking. In *ICLR*.
[3] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. 2020. Adaptive universal generalized pagerank graph neural network. In *ICLR*.
[4] Yash Deshpande, Andrea Montanari, Elchanan Mossel, and Subhabrata Sen. 2018. Contextual stochastic block models. In *NeurIPS*.
[5] Boyuan Feng, Yuke Wang, and Yufei Ding. 2021. UAG: Uncertainty-aware Attention Graph Neural Network for Defending Adversarial Attacks. In *AAAI*.
[6] Fuli Feng, Weiran Huang, Xiangnan He, Xin Xin, Qifan Wang, and Tat-Seng Chua. 2021. Should graph convolution trust neighbors? a simple causal inference method. In *SIGIR*. 1208–1218.
[7] Guoji Fu, Yifan Hou, et al. 2020. Understanding graph neural networks from graph signal denoising perspectives. *arXiv preprint arXiv:2006.04386* (2020).
[8] Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*. PMLR, 1050–1059.
[9] Siavash Golkar, Michael Kagan, and Kyunghyun Cho. 2019. Continual learning via neural pruning. *NeurIPS Workshop Neuro AI* (2019).
[10] Yifan Hou, Jian Zhang, James Cheng, Kaili Ma, Richard TB Ma, Hongzhi Chen, and Ming-Chang Yang. 2020. Measuring and improving the use of graph information in graph neural networks. In *ICLR*.
[11] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open Graph Benchmark: Datasets for Machine Learning on Graphs. *arXiv preprint arXiv:2005.00687* (2020).
[12] Chao Huang, Huance Xu, Yong Xu, Peng Dai, Lianghao Xiao, Mengyin Lu, Liefeng Bo, Hao Xing, Xiaoping Lai, and Yanfang Ye. 2021. Knowledge-aware coupled graph neural network for social recommendation. In *AAAI*.
[13] Mengda Huang, Yang Liu, Xiang Ao, Kuan Li, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. 2022. AUC-oriented Graph Neural Network for Fraud Detection. In *WWW*. 1311–1321.
[14] Tinglin Huang, Yuxiao Dong, Ming Ding, Zhen Yang, Wenzheng Feng, Xinyu Wang, and Jie Tang. 2021. MixGCF: An Improved Training Method for Graph Neural Network-based Recommender Systems. In *KDD*. 665–674.
[15] Yongxiang Huang and Albert CS Chung. 2020. Edge-variational graph convolutional networks for uncertainty-aware disease prediction. In *MICCAI*.
[16] Di Jin, Zhizhi Yu, Cuiying Huo, Rui Wang, Xiao Wang, Dongxiao He, and Jiawei Han. 2021. Universal Graph Convolutional Networks. *NeurIPS* 34 (2021).
[17] Sangwon Jung, Hongjoon Ahn, Sungmin Cha, and Taesup Moon. 2020. Continual learning with node-importance based adaptive group sparse regularization. *NeurIPS* 33 (2020), 3647–3658.
[18] Alex Kendall and Yarin Gal. 2017. What uncertainties do we need in bayesian deep learning for computer vision? *NeurIPS* 30 (2017).
[19] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.
[20] Derek Lim, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalerao, and Ser Nam Lim. 2021. Large Scale Learning on Non-Homophilous Graphs: New Benchmarks and Strong Simple Methods. *NeurIPS* 34 (2021).
[21] Shengchao Liu, Hanchen Wang, Weiyang Liu, Joan Lasenby, et al. 2022. Pre-training Molecular Graph Representation with 3D Geometry. In *ICLR*.
[22] Yang Liu, Xiang Ao, Zidi Qin, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. 2021. Pick and choose: a GNN-based imbalanced learning approach for fraud detection. In *Proceedings of the Web Conference 2021*. 3168–3177.
[23] Miller McPherson, Lynn Smith-Lovin, and James M Cook. 2001. Birds of a feather: Homophily in social networks. *Annual review of sociology* 27, 1 (2001), 415–444.
[24] Hoang Nt and Takanori Maehara. 2019. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550* (2019).
[25] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-gcn: Geometric graph convolutional networks. In *ICLR*.
[26] Everett M Rogers and Dilip K Bhowmik. 1970. Homophily-heterophily: Relational concepts for communication research. *Public opinion quarterly* 34, 4 (1970).
[27] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. 2021. Multi-scale attributed node embedding. *Journal of Complex Networks* 9, 2 (2021), cnab014.
[28] Sebastian Ruder. 2016. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747* (2016).
[29] Jonathan M Stokes, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M Donghia, et al. 2020. A deep learning approach to antibiotic discovery. *Cell* 180, 4 (2020), 688–702.
[30] Otilia Stretcu, Krishnamurthy Viswanathan, et al. 2019. Graph agreement models for semi-supervised learning. In *NeurIPS*.
[31] Susheel Suresh, Vinith Budde, Jennifer Neville, Pan Li, and Jianzhu Ma. 2021. Breaking the Limit of Graph Neural Networks by Improving the Assortativity of Graphs with Local Mixing Patterns. In *KDD*.
[32] Amanda L Traud, Peter J Mucha, and Mason A Porter. 2012. Social structure of Facebook networks. *Physica A: Statistical Mechanics and its Applications* (2012).
[33] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.
[34] Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. 2020. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies* 1, 1 (2020), 396–413.
[35] Minjie Wang and Da Zheng et al. 2019. Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks. *arXiv preprint* (2019).
[36] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation learning on graphs with jumping knowledge networks. In *ICML*. PMLR, 5453–5462.
[37] Li Zhang and Haiping Lu. 2020. A Feature-Importance-Aware and Robust Aggregator for GCN. In *CIKM*. 1813–1822.
[38] Xujiang Zhao, Feng Chen, Shu Hu, and Jin-Hee Cho. 2020. Uncertainty aware semi-supervised learning on graph data. *NeurIPS* 33 (2020), 12827–12836.
[39] Qiwei Zhong, Yang Liu, Xiang Ao, Binbin Hu, Jinghua Feng, Jiayu Tang, and Qing He. 2020. Financial defaulter detection on online credit payment via multi-view attributed heterogeneous information network. In *WWW*. 785–795.
[40] Jiong Zhu, Ryan A Rossi, Anup Rao, Tung Mai, Nedim Lipka, Nesreen K Ahmed, and Danai Koutra. 2021. Graph Neural Networks with Heterophily. In *AAAI*.
[41] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. 2020. Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs. In *NeurIPS*.

A REPRODUCIBILITY

A.1 Datasets

cSBM [4] refers to contextual stochastic block models and has been used for synthetic datasets in [3]. We modify the graph generation process to multi-class version and mix nodes of homophily and heterophily for the experiments in this work.

Penn94 [32] is a friendship network from the Facebook 100 networks of university students from 2005, where nodes represent students. Each node is labeled with the reported gender of the user. The node features are major, second major/minor, dorm/house, year, and high school.

Cora-full [2] is a citation network labeled on the paper topic. Most approaches test the performance on a small subset of this dataset, which has 2708 nodes and strong homophily. We adopt the entire network since it is a typical semi-homophilous graph.

Ogbn-arxiv [11] is the citation network between all Computer Science (CS) arXiv papers indexed by MAG [34]. Each node is an arXiv paper, and each directed edge indicates that one paper cites another one. Each paper comes with a 128-dimensional feature vector obtained by averaging the embeddings of words in its title and abstract. The task is to predict the 40 subject areas of arXiv CS papers, e.g., cs.AI, cs.LG, and cs.OS, which are manually labeled by the paper's authors and arXiv moderators.

Chameleon and **Squirrel** collected by [27] are networks of hyperlinked web pages on Wikipedia related to animal topics. The nodes (here pages) are labelled from one of 5 classes based on the average traffic (views) they received. Node features are bag-of-words representation of nouns in the respective pages.

Wisconsin [25] is collected as part of CMU WebKB project. Nodes are university web pages and edges are hyperlinks between them. Node labels are one of student, project, course, staff or faculty. Node features are bag-of-words representation of the web pages.

Table 4: Statistics of heterophilous graphs.

Dataset	#Node	#Edge	#Class	#Feat
Chameleon	2,277	36,101	5	2,325
Squirrel	5,201	217,073	5	2,089
Wisconsin	251	515	5	1,703

A.2 Compared Methods

- **GCN** [19]: graph convolution network achieved by localized first-order approximation of spectral graph convolutions.
- **GAT** [33]: graph attention network.
- **Mixhop** [1]: repeatedly mixing feature representations of neighbors at various distances.
- **GPR-GNN** [3]: a Generalized PageRank GNN architecture that learns the GPR weights adaptively.
- **JK-Net** [36]: a new aggregation scheme that can adapt neighborhood ranges to nodes individually.
- **H2GCN** [41]: graph convolution network with designs that increase representation power under heterophily.
- **CPGNN** [40]: an approach that models an interpretable class compatibility matrix into GNNs.
- **WRGAT** [31]: an approach that improves the assortativity of graphs with local mixing patterns.
- **U-GCN** [16]: a new universal GCN extracting information from 1-hop, 2-hop and kNN networks simultaneously.

A.3 Implementation Details

UD-GNN is implemented in Pytorch 1.9.0 with Python 3.8, and experiments are running on a Ubuntu 18.04.1 server with 40 cores and 512GB memory. GCN, GAT are implemented based on DGL [35]. JK-Net, Mixhop, GPR-GNN, H2GCN, CPGNN, WAGAT, U-GCN are implemented using the source code provided by the authors.

The train/valid/test split ratio is set to be 40%/ 20%/40% for cSBM, 80%/ 10%/10% for Penn94, and 70%/ 10%/20% for Cora-full. For Ogbn-arxiv, we follow the realistic split provided by the dataset, train on papers published until 2017, validate on those published in 2018, and test on those published since 2019.

A.4 Hyper-parameter Settings

The parameters of UD-GNN are optimized with RMSprop [28] optimizer. Hyper-parameter tuning is conducted using grid search for most methods. In UD-GNN, $N_{\text{epoch}} = 300$, $T = 5$. For cSBM, the debiasing ratio $\gamma = 3 : 2$. For Penn94, the debiasing ratio $\gamma = 1 : 1$. For Cora-full, $\gamma = 7 : 3$. For Ogbn-arxiv, $\gamma = 10 : 7$. The head of attention in GAT is 3. The hop of sampled neighbors in Mixhop is set to be 2. Other hyper-parameters like learning rate (lr) and weight decay (wd) can be referred in Table 5.

Table 5: Hyper-parameters for each dataset.

	cSBM				
	lr	wd	θ	#layers	#hidden
GCN	0.001	0.0005	0.75	3	256
GAT	0.001	0.0005	0.75	3	256
Mixhop	0.001	0.0005	0.75	3	256
GPRGNN	0.001	0.0005	0.75	3	256
	Penn94				
	lr	wd	θ	#layers	#hidden
GCN	0.005	0.01	0.5	2	512
GAT	0.005	0.01	0.5	2	512
Mixhop	0.001	0.005	0.5	2	512
GPRGNN	0.001	0.005	0.5	2	512
	Cora-full				
	lr	wd	θ	#layers	#hidden
GCN	0.001	0.01	0.5	3	1024
GAT	0.001	0.01	0.5	3	1024
Mixhop	0.001	0.01	0.5	3	1024
GPRGNN	0.001	0.01	0.5	3	1024
	Ogbn-arxiv				
	lr	wd	θ	#layers	#hidden
GCN	0.005	0.01	0.5	3	256
GAT	0.005	0.01	0.5	1	256
Mixhop	0.001	0.01	0.5	3	256
GPRGNN	0.001	0.01	0.5	3	256