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

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

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

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.

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

KEYWORDS
Graph Neural Network, Homophily, Uncertainty, Debiased Training

ACM Reference Format:

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 fail 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 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

In this work, we simply set the threshold to be 0.5.
in Figure 1. Cora, Citeseer, and PubMed are typically homophilic graphs where the strong homophily group contains more than 50% nodes. Chameleon, Squirrel, and Wisconsin are heterophilic 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-homophilic graphs.

The performance of GNNs trained on semi-homophilic 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-homophilic 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-homophilic 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 $\mathcal{G} = (\mathcal{V}, \mathcal{A}, \mathcal{X})$, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ is the set of nodes, $\mathcal{X} \in \mathbb{R}^{N \times D}$ denotes node features, and $\mathcal{A} \in \{0, 1\}^{N \times N}$ 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 $Y \in \{0, 1\}^{N \times C}$ where $C$ is the number of classes.

2.1 Homophily Measure

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

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 $h_v = \frac{|E_{\text{intra}}|}{|E_v|}$, where $E_v = \{(u, v) \mid A_{uv} = 1\}$ is the set of edges linked to $v$. 

![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.](image)
We mainly follow the common setting of node classification, where we systematically categorize graph benchmarks from two-level classification loss such as cross-entropy. The parameters are typically learned over a set of labeled nodes by minimizing a classification loss such as cross-entropy.

\[ f(W, A) \rightarrow Y \]  

where \( W \) 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 \cite{gal2016dropout} 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 \( W_b \) with distributions over these parameters, like a Gaussian prior distribution \( W \sim N(0, I) \). Given the graph data \((A, X)\) and the prediction \( Y \), the model likelihood for node classification can be defined as Eq. (2).

\[
p(Y \mid A, X) = \int p(Y \mid W, A, X)p(W \mid A, X)dW \tag{2}
\]

To overcome the intractable inference of evaluating the true model posterior \( p(W \mid A, X) \) in Eq. (2), we utilize Monte Carlo dropout variational inference \cite{gal2016dropout} to approximate it, which defines a simple distribution \( q_b(W) \) by multiplying a deterministic model weight \( W_b \) with a binary mask \( M \) following Bernoulli distribution with parameter \( \theta \), as demonstrated in Eq. (3).
The biased parameters are denoted by $b$. Then the graph neural network is represented by $f$ the uncertainty score of each node. Estimated by Eq. (5), which is a hyper-parameter.

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

$$P(M) \sim \text{Bernoulli}(\theta)$$

$$q_θ(W) \sim M \odot W_b$$

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

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

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.

$$\mathbb{E}[\hat{y} \mid A, X] = \int \hat{y} p(\hat{y} \mid A, X) d\hat{y} \approx \frac{1}{T} \sum_{t=1}^T \hat{y}_t$$

$$U(\hat{y} \mid A, X) = \text{Var}(\hat{y} \mid A, X) \approx \frac{1}{T} \sum_{t=1}^T (\hat{y}_t - \mathbb{E}[\hat{y} \mid A, X])^2$$

3.3 Debiased Training

The biased parameters are denoted by $W_b$, and the well-trained graph neural network is represented by $f_{W_b}$. The purpose of this subsection is to train a debiased graph neural network $f_{W_d}$ from $f_{W_b}$, 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 $V_{\text{train}}$, can be divided into a confident node set $V_{\text{con}}$ and an uncertain node set $V_{\text{unc}}$ according to the uncertainty score $U(\hat{y} \mid A, X_{\text{train}})$ estimated by Eq. (5). The debiasing ratio $\gamma$ is defined to be $\gamma = \frac{||V_{\text{con}}||}{||V_{\text{con}}|| + ||V_{\text{unc}}||}$, where the two sets satisfy $V_{\text{con}} \cup V_{\text{unc}} = V_{\text{train}}$ and $V_{\text{con}} \cap V_{\text{unc}} = \emptyset$.

The confident node set $V_{\text{con}}$ refers to nodes with low uncertainty score. Due to the bias of $W_b$, $f_{W_b}$ has learnt a good representations for $V_{\text{con}}$. The graph neural network $f_{W_b}$ can be pruned by removing unimportant parameters in $W_b$ of which the absolute value is almost zero. $Z$ is a $0 - 1$ pruning mask of $W_b$ with $\theta$ 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 $Z \odot W_b$ on $V_{\text{con}}$ so as to regain the knowledge learnt by the biased model. After this step, there are some free parameters $(1 - Z) \odot W_b$, which will be optimized on $V_{\text{unc}}$.

We simply set the threshold to be $1e-5$ and weights smaller than that would be pruned.
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>
<thead>
<tr>
<th>Dataset</th>
<th>#Node</th>
<th>#Edge</th>
<th>#Class</th>
<th>#Feat</th>
<th>%Heter</th>
</tr>
</thead>
<tbody>
<tr>
<td>cSBM</td>
<td>20,000</td>
<td>998,766</td>
<td>10</td>
<td>1,024</td>
<td>50%</td>
</tr>
<tr>
<td>Penn94</td>
<td>41,554</td>
<td>1,362,229</td>
<td>2</td>
<td>4,814</td>
<td>49%</td>
</tr>
<tr>
<td>Cora-full</td>
<td>19,793</td>
<td>126,842</td>
<td>70</td>
<td>8,710</td>
<td>44%</td>
</tr>
<tr>
<td>Ogbn-arxiv</td>
<td>169,343</td>
<td>1,166,243</td>
<td>40</td>
<td>128</td>
<td>37%</td>
</tr>
</tbody>
</table>

#### 4.1.2 Compared Methods

We compare with several state-of-the-art graph neural network methods, including GCN [19], GAT [33], Mixhop [1], GIR-GNN [3], JK-Net [36], H2GCN [41], CP-GNN [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, GIR-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. The hyper-parameters in UD-GNN include the number of samples $T$, dropout rate $\theta$, and the debiasing ratio $\gamma$. $T$ 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.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>cSBM</th>
<th>Penn94</th>
<th>Cora-full</th>
<th>Ogbn-arxiv</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metric</td>
<td>Acc</td>
<td>δho↓</td>
<td>δhe↓</td>
<td>Acc↑ δho↓</td>
</tr>
<tr>
<td>GCN</td>
<td>VA</td>
<td>47.30±0.43</td>
<td>1.23</td>
<td>9.11</td>
</tr>
<tr>
<td></td>
<td>UD</td>
<td>49.39±0.16</td>
<td>2.31</td>
<td>4.42</td>
</tr>
<tr>
<td>GAT</td>
<td>VA</td>
<td>50.44±0.29</td>
<td>1.68</td>
<td>8.27</td>
</tr>
<tr>
<td></td>
<td>UD</td>
<td>52.66±0.31</td>
<td>2.04</td>
<td>3.12</td>
</tr>
<tr>
<td>Mixhop</td>
<td>VA</td>
<td>51.31±0.41</td>
<td>0.93</td>
<td>8.25</td>
</tr>
<tr>
<td></td>
<td>UD</td>
<td>52.51±0.17</td>
<td>1.84</td>
<td>3.17</td>
</tr>
<tr>
<td>GPRGNN</td>
<td>VA</td>
<td>53.79±0.15</td>
<td>1.29</td>
<td>9.23</td>
</tr>
<tr>
<td></td>
<td>UD</td>
<td>54.80±0.20</td>
<td>1.62</td>
<td>2.48</td>
</tr>
</tbody>
</table>

UD, i.e. the sum of $\delta_{ho}$ and $\delta_{he}$, 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.

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 $\delta$. 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.

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

To answer Q2, 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.
Since our main focus is the bias issue, we only report the relative bias and denote the sum of $\delta_{ho}$ and $\delta_{he}$ as $\delta$ below for brevity. The full model UD-GNN always achieves the lowest $\delta$ 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 $\delta$ 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 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, 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 $\delta_{ho}$ and $\delta_{he}$ as $\delta$ below for brevity. The full model UD-GNN always achieves the lowest $\delta$ 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 $\delta$ 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.

### 4.5 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 $\delta$. 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.

### 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 $\delta_{ho}$ and $\delta_{he}$ is denoted as $\delta$ 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 $\delta$ 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.

![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 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.](image-url)
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 $\delta$ is near 0. As the number of classes grows larger than 5, the accuracy declines, and the $\delta$ 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. CP-GNN [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


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 re trains 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


[5] Boyuan Feng, Yue Wang, and Yufei Ding. 2021. UAG: Uncertainty-aware Atten-


UD-GNN: Uncertainty-aware Debiased Training on Semi-Homophilous Graphs KDD ’22, August 14–18, 2022, Washington, DC, USA
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.

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.
- CP-GNN [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, CP-GNN, 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, Nepoch = 500, T = 5. For cSBM, the debiasing ratio γ = 3 : 2. For Penn94, the debiasing ratio γ = 1 : 1. For Cora-full, γ = 7 : 3. For Ogbn-arxiv, γ = 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>
<thead>
<tr>
<th>Dataset</th>
<th>#Node</th>
<th>#Edge</th>
<th>#Class</th>
<th>#Feat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chameleon</td>
<td>2,277</td>
<td>36,101</td>
<td>5</td>
<td>2,325</td>
</tr>
<tr>
<td>Squirrel</td>
<td>5,201</td>
<td>217,073</td>
<td>5</td>
<td>2,089</td>
</tr>
<tr>
<td>Wisconsin</td>
<td>251</td>
<td>515</td>
<td>5</td>
<td>1,703</td>
</tr>
</tbody>
</table>

Table 4: Statistics of heterophilous graphs.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>lr</th>
<th>wd</th>
<th>θ</th>
<th>#layers</th>
<th>#hidden</th>
</tr>
</thead>
<tbody>
<tr>
<td>cSBM</td>
<td>GCN</td>
<td>0.001</td>
<td>0.0005</td>
<td>0.75</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>GAT</td>
<td>0.001</td>
<td>0.0005</td>
<td>0.75</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Mixhop</td>
<td>0.001</td>
<td>0.0005</td>
<td>0.75</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>GPR-GNN</td>
<td>0.001</td>
<td>0.0005</td>
<td>0.75</td>
<td>3</td>
</tr>
<tr>
<td>Penn94</td>
<td>GCN</td>
<td>0.005</td>
<td>0.01</td>
<td>0.5</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>GAT</td>
<td>0.005</td>
<td>0.01</td>
<td>0.5</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Mixhop</td>
<td>0.001</td>
<td>0.005</td>
<td>0.5</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>GPR-GNN</td>
<td>0.001</td>
<td>0.005</td>
<td>0.5</td>
<td>2</td>
</tr>
<tr>
<td>Cora-full</td>
<td>GCN</td>
<td>0.001</td>
<td>0.01</td>
<td>0.5</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>GAT</td>
<td>0.001</td>
<td>0.01</td>
<td>0.5</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Mixhop</td>
<td>0.001</td>
<td>0.01</td>
<td>0.5</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>GPR-GNN</td>
<td>0.001</td>
<td>0.01</td>
<td>0.5</td>
<td>3</td>
</tr>
<tr>
<td>Ogbn-arxiv</td>
<td>GCN</td>
<td>0.005</td>
<td>0.01</td>
<td>0.5</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>GAT</td>
<td>0.005</td>
<td>0.01</td>
<td>0.5</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Mixhop</td>
<td>0.001</td>
<td>0.01</td>
<td>0.5</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>GPR-GNN</td>
<td>0.001</td>
<td>0.01</td>
<td>0.5</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 5: Hyper-parameters for each dataset.