Stabilizing GNN for Fairness via Lipschitz Bounds

Anonymous Authors[1]

Abstract

The Lipschitz bound, a technique from robust statistics, limits the maximum changes in output with respect to the input, considering associated irrelevant biased factors. It provides an efficient and provable method for examining the output stability of machine learning models without incurring additional computation costs. However, there has been no previous research investigating the Lipschitz bounds for Graph Neural Networks (GNNs), especially in the context of non-Euclidean data with inherent biases. This poses a challenge for constraining GNN output perturbations induced by input biases and ensuring fairness during training. This paper addresses this gap by formulating a Lipschitz bound for GNNs operating on attributed graphs, and analyzing how the Lipschitz constant can constrain output perturbations induced by biases for fairness training. The effectiveness of the Lipschitz bound is experimentally validated in limiting model output biases. Additionally, from a training dynamics perspective, we demonstrate how the theoretical Lipschitz bound can effectively guide GNN training to balance accuracy and fairness.

1. Introduction

Graphs, as non-Euclidean data, are widely used in various real-world applications, such as recommender systems (Shalaby et al., 2017; Huang et al., 2021; Li et al., 2021), drug discovery (Takigawa & Mamitsuka, 2013; Li et al., 2017), and knowledge engineering (Rizun, 2019; Wang et al., 2018). Learning on non-Euclidean data has led to the development of Graph Neural Networks (GNNs) combined with deep learning (Gori et al., 2005; Scarselli et al., 2005; Li et al., 2016; Hamilton et al., 2017; Xu et al., 2019). Graph Convolutional Networks (GCNs) (Kipf & Welling, 2017; Zhang & Chen, 2018; Fan et al., 2019) are the most referenced GNN architecture, utilizing convolutional layers and message-passing mechanisms for graph learning.

In parallel with the successful applications of GNNs in various scenarios, there is an increasing societal concern regarding the interpretability of GNN models and their interactions with graph data during training (Dong et al., 2021; Lahoti et al., 2019b; Kang et al., 2020; Mujkanovic et al., 2022). Existing works often lack a clear understanding of how biases learned from the input affect output stability and fairness considerations (Dong et al., 2021; Kang et al., 2020; Liao et al., 2021; Li et al., 2021). In the light of this, we aim to constrain the unwanted changes in GNN output, particularly when the training graph data contains hidden biases. Consequently, our fundamental research question is:

Without extra computations, how can we constrain the unwanted changes in GNN output, particularly when the graph training data has hidden learnable biases?

This question is motivated by the need to control the model's sensitivity to unfair correlations caused by irrelevant factors in the training process. By constraining output perturbations during training, we can improve GNN generalization, mitigate unfair biases induced by data, and enhance the model's resilience against perturbations.

To answer this question, we propose the use of Lipschitz bounds, which are commonly used in robust statistics to study the maximum possible changes in output due to irrelevant factors in the input. We introduce Lipschitz bounds to GNN training, providing an interpretable solution to ensure fairness. Specifically, we derive the Lipschitz bound for general GNNs and analyze the GNN's predictions in relation to input biases, focusing on individual fairness from a ranking perspective. By constraining unfair changes, we stabilize GNN outputs. Furthermore, we facilitate the calculation of Lipschitz bounds by deriving the Jacobian matrix of the model, allowing for practical training approximations. Our theoretical analysis of Lipschitz bounds guides the implementation of GNN fairness training, which is validated through experiments on diverse datasets and various graph tasks. In summary, our contributions are as follows:

- We derive the Lipschitz bound for general GNNs and use it to constrain unfair changes in predictions induced by input biases, particularly for individual fairness from a

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the SCIS workshop.

ranking perspective.

- We facilitate the calculation of Lipschitz bounds by deriving the Jacobian matrix of the model, enabling practical training approximations.
- We demonstrate through experiments that our theoretical analysis of Lipschitz bounds effectively guides GNN fairness training. Our solution is plug-and-play, compatible with different existing methods, and applicable to diverse datasets and graph tasks.

2. Estimating Lipschitz Bounds for GNNs with Fairness Considerations

In this section, we focus on estimating the upper bounds of the Lipschitz constants for GNNs, which are crucial for analyzing output perturbations induced by input biases. We begin by establishing Lipschitz bounds for GNNs, providing closed-form formulas for these bounds. Then, we derive the Jacobian matrix of the GNN model to facilitate the calculation of Lipschitz bounds in practice. Finally, we demonstrate how Lipschitz bounds can be used to promote individual fairness, ensuring output consistency according to the rank-based individual fairness definition.

2.1. Stability of Model Output

A GNN is a function that transforms graph features from the input space to the output space. Formally, a GNN can be represented as $f : \boldsymbol{X} \in \mathbb{R}^{N \times F^{\text{in}}} \to \boldsymbol{Y} \in \mathbb{R}^{N \times F^{\text{out}}}$, where $\boldsymbol{X}$ is the input feature matrix, $\boldsymbol{Y}$ is the output feature matrix, and N is the number of nodes in the graph.

To analyze the stability of the model's output, we examine the Lipschitz bound of the Jacobian matrix of the GNN model. For this purpose, we introduce the following lemma, which establishes an inequality relation:

Lemma 1. *For any vectors x and y in a Euclidean space, let $g(x)$ and $g(y)$ be vector-valued functions of x and y, respectively, and let g_i be the i-th component function of g. Then, the following inequality holds:*

$$\frac{\|g(x) - g(y)\|}{\|x - y\|} \leqslant \left\| \left[\frac{g_i(x) - g_i(y)}{\|x - y\|} \right]_{i=1}^{n} \right\|, \tag{1}$$

where $\| \cdot \|$ represents the norm of a vector.

Lemma 1 provides an inequality that relates the norm of the difference between two vector-valued functions, $g(x)$ and $g(y)$, to the norm of a vector composed of the component-wise differences of the functions evaluated at x and y. Based on Lemma 1, we can now present the following theorem:

Theorem 1. *Let $\boldsymbol{Y}$ be the output of an L-layer GNN represented by $f(\cdot)$ with $\boldsymbol{X}$ as the input. Assuming that the activation function, represented by $\rho(\cdot)$, is ReLU with a Lipschitz constant of $\text{Lip}(\rho) = 1$, the global Lipschitz constant of the GNN, denoted as $\text{Lip}(f)$, satisfies:*

$$\text{Lip}(f) \leqslant \max_j \prod_{l=1}^{L} \left\| F^{l'} \right\| \left\| \left[\mathcal{J}(h^l) \right]_j \right\|_\infty, \tag{2}$$

where $F^{l'}$ represents the output dimension of the l-th message-passing layer, j is the index of the node (e.g., j-th), and the vector $\left[\mathcal{J}(h^l) \right] = \left[\left\| \boldsymbol{J}_1(h^l) \right\|, \left\| \boldsymbol{J}_2(h^l) \right\|, \cdots, \left\| \boldsymbol{J}_{F^{l'}}(h^l) \right\| \right]$. Notably, $\boldsymbol{J}_i(h^l)$ denotes the i-th row of the Jacobian matrix of the l-th layer's input and output, and $\left[\mathcal{J}(h^l) \right]_j$ is the vector corresponding to the j-th node in the l-th layer $h^l(\cdot)$.

Please refer to the Appendix C for the complete proof.

2.2. Simplifying Bounds Calculation by Jacobian Matrix

The approach presented in Theorem 1 for estimating the Lipschitz constants $\text{Lip}(f)$ across different layers of the GNN $f(\cdot)$ requires explicit expressions for each component, making the process somewhat challenging. However, this difficulty can be mitigated by computing the corresponding Jacobian matrices, as shown in Equation (4) of the paper. By leveraging the values of each component in the Jacobian matrix, we can approximate the Lipschitz constants in a straightforward manner. Additionally, the expression $\prod_{l=1}^{L} \|F^{l'}\| \|\mathcal{J}(\mathbf{h}^l)_j\|_\infty$ provides valuable insights into the factors that influence the Lipschitz bounds, such as the dimensions of the output layer and the depth of the GNN.

However, considering the potential presence of multiple hierarchical layers in the network, their cumulative effect could lead to a significant deviation from the original bounds. Therefore, it is beneficial to consider the entire network as a single layer and directly derive the Lipschitz bound from the input to the output. To achieve this, we introduce the Jacobian matrix in detail. Let $\boldsymbol{J}_i$ denote the Jacobian matrix of the i-th node, which can be calculated as:

$$\boldsymbol{J}_i = \begin{bmatrix} \frac{\partial \boldsymbol{Y}_{i1}}{\partial \boldsymbol{X}_{i1}} & \frac{\partial \boldsymbol{Y}_{i1}}{\partial \boldsymbol{X}_{i2}} & \cdots & \frac{\partial \boldsymbol{Y}_{i1}}{\partial \boldsymbol{X}_{iF^{\text{in}}}} \\ \frac{\partial \boldsymbol{Y}_{i2}}{\partial \boldsymbol{X}_{i1}} & \frac{\partial \boldsymbol{Y}_{i2}}{\partial \boldsymbol{X}_{i2}} & \cdots & \frac{\partial \boldsymbol{Y}_{i2}}{\partial \boldsymbol{X}_{iF^{\text{in}}}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \boldsymbol{Y}_{iF^{\text{out}}}}{\partial \boldsymbol{X}_{i1}} & \frac{\partial \boldsymbol{Y}_{iF^{\text{out}}}}{\partial \boldsymbol{X}_{i2}} & \cdots & \frac{\partial \boldsymbol{Y}_{iF^{\text{out}}}}{\partial \boldsymbol{X}_{iF^{\text{in}}}} \end{bmatrix}_{F^{\text{out}} \times F^{\text{in}}}. \tag{3}$$

We can define $\boldsymbol{J}_i = \left[\boldsymbol{J}_{i1}^\top, \boldsymbol{J}_{i2}^\top, \ldots, \boldsymbol{J}_{iF^{\text{out}}}^\top \right]^\top$, where $\boldsymbol{J}_{ij} = \left[\frac{\partial \boldsymbol{Y}_{ij}}{\partial \boldsymbol{X}_{i1}}, \frac{\partial \boldsymbol{Y}_{ij}}{\partial \boldsymbol{X}_{i2}}, \ldots, \frac{\partial \boldsymbol{Y}_{ij}}{\partial \boldsymbol{X}_{iF^{\text{in}}}} \right]^\top$. Then, we define $\mathcal{J}_i = [\|\boldsymbol{J}_{i1}\|, \|\boldsymbol{J}_{i2}\|, \ldots, \|\boldsymbol{J}_{iF^{\text{out}}}\|]^\top$ and $\mathcal{J} = \left[\mathcal{J}_1^\top, \mathcal{J}_2^\top, \ldots, \mathcal{J}_N^\top \right]^\top$. To analyze the Lipschitz bounds of the Jacobian matrices of all output features for N nodes, we define $\mathrm{LB}(\mathcal{J})$ as follows:

$$\mathrm{LB}(\mathcal{J}) = \begin{bmatrix} \mathcal{J}_1^\top \\ \mathcal{J}_2^\top \\ \vdots \\ \mathcal{J}_N^\top \end{bmatrix} = \begin{bmatrix} \mathcal{J}_{11} & \mathcal{J}_{12} & \cdots & \mathcal{J}_{1F^{\text{out}}} \\ \mathcal{J}_{21} & \mathcal{J}_{22} & \cdots & \mathcal{J}_{2F^{\text{out}}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{J}_{N1} & \mathcal{J}_{N2} & \cdots & \mathcal{J}_{NF^{\text{out}}} \end{bmatrix}_{N \times F^{\text{out}}}, \tag{4}$$

where $\mathcal{J}_{ij} = \|\boldsymbol{J}_{ij}\|$. Based on the definition of $\mathrm{LB}(\mathcal{J})$, we can establish the Lipschitz bound of the entire GNN model during training in the next subsection. To measure the scale of $\mathrm{LB}(\mathcal{J})$ of GNN $f(\cdot)$, we define $\mathrm{Lip}(f)$ as follows:

$$\mathrm{Lip}(f) = \|\mathrm{LB}(\mathcal{J})\|_{\infty,2}, \tag{5}$$

where $\|\cdot\|_{\infty,2}$ denotes the element-wise l_2-norm of the rows of $\mathrm{LB}(\mathcal{J})$ and then takes the maximum norm among all rows. This calculation involves taking the l_2-norm for each row of $\mathrm{LB}(\mathcal{J})$ and then taking the maximum norm among all rows. Therefore, we have proposed an easy solution to approximate $\prod_{l=1}^{L} \|F^{l\prime}\| \|\mathcal{J}(\mathbf{h}^l)_j\|$ for *practical training*.

2.3. Promoting GNN Fairness in Practice

By regularizing GNN training with input-output rank consistency, especially in the context of rank-based individual fairness, we can leverage Lipschitz bounds to alleviate biases inherent in training data and promote individual fairness. The Lipschitz bounds, denoted as $\mathrm{Lip}(f) = \max_j \prod_{l=1}^{L} \|F^{l\prime}\| \|\mathcal{J}(\mathbf{h}^l)_j\|_\infty$, ensure that the output predictions of the model remain consistent with the ranking lists based on the similarity of each node in the input graph to other nodes in the oracle pairwise similarity matrix $\boldsymbol{S_G}$ and the similarity matrix of the predicted outcome space $\boldsymbol{S_Y}$, as defined in the rank-based individual fairness of GNNs.

To facilitate the integration of Lipschitz bounds into existing fairness-oriented GNN training processes, we introduce a plug-and-play solution called JacoLip. Algorithm 1 in the Appendix displays a detailed PyTorch-style pseudocode of this solution. It involves training the GNN model with Lipschitz bounds for a predetermined number of epochs. Throughout the training phase, the Lipschitz constant for the model's output is computed using the gradients and norms of the input features. This Lipschitz bound serves as a regularization term within the loss function, ensuring that the model's output stays within defined constraints.

3. Experiments

In this section, we conducted two major experiments to examine the Lipschitz bound we analyzed in the previous section: *(1)* we utilize Lipschitz constants to bound GNN output consistency for increasing rank-based individual fairness on node classification and link prediction tasks; *(2)* we then validate the constraint effects of Lipschitz bounds on GNN gradients/weights with regards to biases induced by data from training dynamics perspective (Appendix F).

3.1. Setup

Datasets. We evaluate the Lipschitz bound's effectiveness in promoting individual fairness in GNNs on real-world datasets, including citation networks (*ACM*), co-authorship networks (*Co-author-CS* and *Co-author-Phy*), and social networks (*BlogCatalog*, *Flickr*, and *Facebook*).

Backbones. We use two widely-used GNN architectures, GCN and SGC, for the node classification task, and GCN and GAE for the link prediction task.

Baselines. In the previous work on rank-based individual fairness (Dong et al., 2021), existing group fairness graph embedding methods, such as (Bose & Hamilton, 2019; Rahman et al., 2019a), are unsuitable for comparison as they promote fairness for subgroups determined by specific protected attributes, whereas our focus is on individual fairness without such attributes. To evaluate our proposed method, we compare it with three important baselines for rank-based individual fairness: *Redress* (Dong et al., 2021), *InFoRM* (Kang et al., 2020), and *PFR* (Lahoti et al., 2019b). Details about these baselines can be found in the Appendix.

Evaluation Metrics We use accuracy (*Acc.*) for node classification, area under the ROC curve (*AUC*) for link prediction, and NDCG@10 for individual fairness evaluation.

Implementation Details Experiments are implemented in PyTorch using released implementations of GNN backbones. The learning rate is set to 0.01, and model-specific hyperparameters are provided in the Appendix. We optimize models using the Adam optimizer, and dataset splitting details are included in the Appendix.

3.2. Effect of Lipschitz Bound to Fairness on Graphs

The experiments conducted on real-world graphs demonstrate the effectiveness of the Lipschitz bound in promoting individual fairness in GNNs from a ranking perspective. The results are summarized in Tables 1 for node classification.

Overall, in Table 1, our JacoLip shows promising results in achieving a better trade-off between accuracy and fairness compared to the baselines: *(i)* when applied to Vanilla models (GCN or SGC), JacoLip improves the fairness performance (NDCG@10) while maintaining comparable accuracy. This demonstrates that optimizing common GNN backbones with the plug-and-play Lipschitz bounds regularization successfully constrains bias during training and promotes a better trade-off between accuracy and fairness; *(ii)* furthermore, when applied to the existing fairness-oriented algorithm Redress, a competitive rank-based individual fairness method, JacoLip with Lipschitz bounds regularization helps to constrain irrelevant biased factors during training and marginally improves the trade-off between accuracy and

Table 1: Evaluation on node classification task: comparing under accuracy and NDCG. Higher performance in both metrics indicates better trade-off. Results are in percentages, and averaged values and standard deviations are computed from five runs. The improvement is within brackets.

Data	Model	Fair Alg.	Feature Similarity		Structural Similarity	
			utility: Acc.↑	fairness: NDCG@10↑	utility: Acc.↑	fairness: NDCG@10↑
ACM	GCN	Vanilla (Kipf & Welling, 2017)	72.49±0.6	47.33±1.0	72.49±0.6	25.42±0.6
		InFoRM (Kang et al., 2020)	68.03±0.3(−6.15%)	39.79±0.3(−15.9%)	69.13±0.5(−4.64%)	12.02±0.4(−52.7%)
		PFR (Lahoti et al., 2019b)	67.88±1.1(−6.36%)	31.20±0.2(−34.1%)	69.00±0.7(−4.81%)	23.85±1.3(−6.18%)
		Redress (Dong et al., 2021)	71.75±0.4(−1.02%)	49.13±0.4(+3.80%)	72.03±0.9(−0.63%)	29.09±0.4(+14.4%)
		JacoLip (on Vanilla)	72.37±0.3(−0.16%)	49.80±0.3(+5.26%)	71.97±0.3(−0.71%)	27.91±0.7(+9.79%)
		JacoLip (on Redress)	71.92±0.2(−0.78%)	53.62±0.6(+13.3%)	72.05±0.5(−0.60%)	31.80±0.4(+25.1%)
	SGC	Vanilla (Wu et al., 2019)	68.40±1.0	55.75±1.1	68.40±1.0	37.18±0.6
		InFoRM (Kang et al., 2020)	68.81±0.5(+0.60%)	48.25±0.5(−13.5%)	66.71±0.6(−2.47%)	28.33±0.6(−23.8%)
		PFR (Lahoti et al., 2019b)	67.97±0.7(−0.62%)	34.71±0.1(−37.7%)	67.78±0.1(−0.91%)	37.15±0.6(−0.08%)
		Redress (Dong et al., 2021)	67.16±0.2(−1.81%)	58.64±0.4(+5.18%)	67.77±0.4(−0.92%)	38.95±0.1(+4.76%)
		JacoLip (on Vanilla)	73.84±0.2(+7.95%)	62.00±0.2(+11.21%)	69.28±0.3(+1.29%)	38.36±0.4(+3.17%)
		JacoLip (on Redress)	72.36±0.4(+5.79%)	69.22±0.5(+24.16%)	72.52±0.5(+6.02%)	41.07±0.3(+10.5%)
CS	GCN	Vanilla (Kipf & Welling, 2017)	90.59±0.3	50.84±1.2	90.59±0.3	18.29±0.8
		InFoRM (Kang et al., 2020)	88.66±1.1(−2.13%)	53.38±1.6(+5.00%)	87.55±0.9(−3.36%)	19.18±0.9(+4.87%)
		PFR (Lahoti et al., 2019b)	87.51±0.7(−3.40%)	37.12±0.9(−27.0%)	86.16±0.2(−4.89%)	11.98±1.3(−34.5%)
		Redress (Dong et al., 2021)	90.70±0.2(+0.12%)	55.01±1.9(+8.20%)	89.16±0.3(−1.58%)	21.28±0.3(+16.4%)
		JacoLip (on Vanilla)	90.68±0.3(+0.90%)	55.35±0.2(+8.87%)	89.23±0.5(−1.50%)	21.82±0.2(+19.3%)
		JacoLip (on Redress)	90.63±0.3(+0.40%)	68.20±0.4(+34.2%)	89.21±0.1(−1.52%)	31.82±0.4(+74.1%)
	SGC	Vanilla (Wu et al., 2019)	87.48±0.8	74.00±0.1	87.48±0.8	32.36±0.3
		InFoRM (Kang et al., 2020)	88.07±0.1(+0.67%)	74.29±0.1(+0.39%)	88.65±0.4(+1.34%)	32.37±0.4(+0.03%)
		PFR (Lahoti et al., 2019b)	88.31±0.1(+0.94%)	48.40±0.1(−34.6%)	84.34±0.3(−3.59%)	28.87±0.9(−10.8%)
		Redress (Dong et al., 2021)	90.01±0.2(+2.89%)	76.60±0.1(+3.51%)	89.35±0.1(+2.14%)	34.24±0.2(+5.81%)
		JacoLip (on Vanilla)	90.23±0.2(+3.14%)	74.63±0.2(+0.85%)	89.53±0.6(+2.34%)	32.83±0.3(+1.45%)
		JacoLip (on Redress)	90.12±0.3(+3.02%)	77.01±0.1(+4.07%)	89.80±0.2(+2.65%)	34.89±0.5(+7.82%)
Phy	GCN	Vanilla (Kipf & Welling, 2017)	94.81±0.2	34.83±1.1	94.81±0.2	1.57±0.1
		InFoRM (Kang et al., 2020)	89.33±0.8(−5.78%)	31.25±0.0(−10.3%)	94.46±0.2(−0.37%)	1.77±0.0(+12.7%)
		PFR (Lahoti et al., 2019b)	89.74±0.5(−5.35%)	24.16±0.4(−30.6%)	87.26±0.2(−7.96%)	1.20±0.1(−23.6%)
		Redress (Dong et al., 2021)	94.63±0.7(−0.19%)	43.64±0.5(+25.3%)	93.94±0.3(−0.92%)	1.93±0.1(+22.9%)
		JacoLip (on Vanilla)	94.60±0.2(−0.22%)	37.33±0.5(+7.18%)	93.99±0.4(−0.86%)	1.87±0.2(+19.1%)
		JacoLip (on Redress)	94.50±0.2(−0.32%)	49.37±0.3(+4.17%)	93.86±0.9(−1.00%)	2.72±0.1(+73.3%)
	SGC	Vanilla (Wu et al., 2019)	94.45±0.2	49.63±0.1	94.45±0.2	3.61±0.1
		InFoRM (Kang et al., 2020)	92.01±0.1(−2.58%)	43.87±0.2(−11.6%)	94.27±0.3(−0.19%)	3.64±0.0(+0.83%)
		PFR (Lahoti et al., 2019b)	89.74±0.3(−4.99%)	28.54±0.1(−42.5%)	89.73±0.3(−5.00%)	2.62±0.1(−27.4%)
		Redress (Dong et al., 2021)	94.30±0.1(−0.16%)	53.40±0.1(+7.60%)	93.94±0.2(−0.54%)	4.03±0.0(+11.6%)
		JacoLip (on Vanilla)	94.20±0.3(−0.26%)	50.70±0.4(+2.16%)	93.58±0.2(−0.92%)	3.80±0.6(+5.26%)
		JacoLip (on Redress)	93.28±0.1(−1.24%)	59.20±0.6(+19.3%)	93.99±1.1(−0.49%)	4.30±0.5(+19.1%)

fairness across different datasets and backbones.

4. Conclusions

We have investigated the use of Lipschitz bounds to promote individual fairness in GNNs from a ranking perspective. We conduct a thorough analysis of the theoretical properties of Lipschitz bounds and their relationship to rank-based individual fairness. Building upon this analysis, we propose JacoLip, a Lipschitz-based fairness solution that incorporates Lipschitz bound regularization into the training process of GNNs. To evaluate the effectiveness of JacoLip, extensive experiments are conducted on real-world datasets for node classification and link prediction. The results consistently demonstrate that JacoLip effectively constrains biased factors during training, leading to improved fairness performance while maintaining accuracy.

In addition to the findings presented in the main paper, we provide further details in the appendix. The appendix contains related work about graph learning, as well as Lipschitz bounds in deep models (Appendix A). We also provide a detailed explanation of preliminary such as Lipschitz constant, GNN, and rank-based individual fairness (Appendix B). In Appendix C, we include the complete proof of Lemma 1 and Theorem 1. Additionally, Appendix D and Appendix E provide detailed descriptions of our model, datasets, and implementations. Furthermore, in Appendix F, we include additional exploration experiments conducted on various datasets and GNN architectures to analyze the training dynamics of JacoLip. Overall, the appendix complements the main paper by providing supplementary information and comprehensive experimental results.

References

Araujo, A., Negrevergne, B., Chevaleyre, Y., and Atif, J. On Lipschitz regularization of convolutional layers using toeplitz matrix theory. In *AAAI Conference on Artificial Intelligence*, 2021. 7

Bose, A. and Hamilton, W. Compositional fairness constraints for graph embeddings. In *International Conference on Machine Learning*, 2019. 3, 7, 10

Buyl, M. and De Bie, T. Debayes: a bayesian method for debiasing network embeddings. In *International Conference on Machine Learning*, 2020. 7

Chen, Z., Li, P., Liu, H., and Hong, P. Characterizing the influence of graph elements. In *International Conference on Learning Representations*, 2023. 7

Dai, E. and Wang, S. Say no to the discrimination: Learning fair graph neural networks with limited sensitive attribute information. In *ACM International Conference on Web Search and Data Mining*, 2021. 7

Dasoulas, G., Scaman, K., and Virmaux, A. Lipschitz normalization for self-attention layers with application to graph neural networks. In *International Conference on Machine Learning*, 2021. 7

Dong, Y., Kang, J., Tong, H., and Li, J. Individual fairness for graph neural networks: A ranking based approach. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2021. 1, 3, 4, 7, 9, 10, 11, 13, 14

Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., and Yin, D. Graph neural networks for social recommendation. In *International World Wide Web Conference*, 2019. 1

Gama, F. and Sojoudi, S. Distributed linear-quadratic control with graph neural networks. *Signal Processing*, 196: 108506, 2022. 7

Gori, M., Monfardini, G., and Scarselli, F. A new model for learning in graph domains. In *IEEE International Joint Conference on Neural Networks*, 2005. 1

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, 2017. 1

Hardt, M., Price, E., Price, E., and Srebro, N. Equality of opportunity in supervised learning. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, 2016. 10

Huang, C., Chen, J., Xia, L., Xu, Y., Dai, P., Chen, Y., Bo, L., Zhao, J., and Huang, J. X. Graph-enhanced multi-task learning of multi-level transition dynamics for session-based recommendation. In *AAAI Conference on Artificial Intelligence*, 2021. 1

Huang, Q., Yamada, M., Tian, Y., Singh, D., and Chang, Y. Graphlime: Local interpretable model explanations for graph neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 2022. 7

Huang, X., Li, J., and Hu, X. Label informed attributed network embedding. In *ACM International Conference on Web Search and Data Mining*, 2017. 9

Järvelin, K. and Kekäläinen, J. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002. 11

Kang, J., He, J., Maciejewski, R., and Tong, H. Inform: Individual fairness on graph mining. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2020. 1, 3, 4, 10, 13, 14

Kim, H., Papamakarios, G., and Mnih, A. The Lipschitz constant of self-attention. In *International Conference on Machine Learning*, 2021. 7

Kipf, T. N. and Welling, M. Variational graph auto-encoders. In *NIPS Workshop on Bayesian Deep Learning*, 2016. 10, 14

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. 1, 4, 10, 13, 14

Lahoti, P., Gummadi, K. P., and Weikum, G. ifair: Learning individually fair data representations for algorithmic decision making. In *International Conference on Data Engineering*, 2019a. 10

Lahoti, P., Gummadi, K. P., and Weikum, G. Operationalizing individual fairness with pairwise fair representations. In *VLDB Endowment*, 2019b. 1, 3, 4, 10, 13, 14

Leskovec, J. and Mcauley, J. Learning to discover social circles in ego networks. In *Advances in Neural Information Processing Systems*, 2012. 9

Li, J., Cai, D., and He, X. Learning graph-level representation for drug discovery. *arXiv preprint arXiv:1709.03741*, 2017. 1

Li, P., Wang, Y., Zhao, H., Hong, P., and Liu, H. On dyadic fairness: Exploring and mitigating bias in graph connections. In *International Conference on Learning Representations*, 2021. 1

Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R. Gated graph sequence neural networks. *International Conference on Learning Representations*, 2016. 1

Liao, P., Zhao, H., Xu, K., Jaakkola, T., Gordon, G. J., Jegelka, S., and Salakhutdinov, R. Information obfuscation of graph neural networks. In *International Conference on Machine Learning*, 2021. 1, 7

Mujkanovic, F., Geisler, S., Günnemann, S., and Bojchevski, A. Are defenses for graph neural networks robust? In *Advances in Neural Information Processing Systems*, 2022. 1

Palowitch, J. J. and Perozzi, B. Debiasing graph embeddings with metadata-orthogonal training. In *Advances in Social Network Analysis and Mining*, 2020. 7

Rahman, T., Surma, B., Backes, M., and Zhang, Y. Fairwalk: Towards fair graph embedding. In *International Joint Conference on Artificial Intelligence*, pp. 3289–3295, 2019a. 3, 10

Rahman, T., Surma, B., Backes, M., and Zhang, Y. Fairwalk: Towards fair graph embedding. In *International Joint Conference on Artificial Intelligence*, 2019b. 7

Rizun, M. Knowledge graph application in education: a literature review. *Acta Universitatis Lodziensis. Folia Oeconomica*, 2019. 1

Scarselli, F., Yong, S. L., Gori, M., Hagenbuchner, M., Tsoi, A. C., and Maggini, M. Graph neural networks for ranking web pages. In *IEEE/WIC/ACM International Conference on Web Intelligence*, 2005. 1

Shalaby, W., AlAila, B., Korayem, M., Pournajaf, L., Al-Jadda, K., Quinn, S., and Zadrozny, W. Help me find a job: A graph-based approach for job recommendation at scale. In *IEEE International Conference on Big Data*, 2017. 1

Shchur, O., Mumme, M., Bojchevski, A., and Günnemann, S. Pitfalls of graph neural network evaluation. In *Relational Representation Learning Workshop, NeurIPS*, 2018. 9

Takigawa, I. and Mamitsuka, H. Graph mining: procedure, application to drug discovery and recent advances. *Drug Discovery Today*, 2013. 1

Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., and Su, Z. Arnetminer: extraction and mining of academic social networks. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008. 9

Tang, L. and Liu, H. Relational learning via latent social dimensions. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2009. 9

Terris, M., Repetti, A., Pesquet, J.-C., and Wiaux, Y. Building firmly nonexpansive convolutional neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2020. 7

Wang, R., Yan, Y., Wang, J., Jia, Y., Zhang, Y., Zhang, W., and Wang, X. Acekg: A large-scale knowledge graph for academic data mining. In *ACM International Conference on Information and Knowledge Management*, 2018. 1

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. Simplifying graph convolutional networks. In *International Conference on Machine Learning*, 2019. 4, 10, 13

Wu, L., Chen, L., Shao, P., Hong, R., Wang, X., and Wang, M. Learning fair representations for recommendation: A graph-based perspective. In *International World Wide Web Conference*, 2021. 7

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. 1

Ying, Z., Bourgeois, D., You, J., Zitnik, M., and Leskovec, J. Gnnexplainer: Generating explanations for graph neural networks. In *Advances in Neural Information Processing Systems*, 2019. 7

Yuan, H., Yu, H., Wang, J., Li, K., and Ji, S. On explainability of graph neural networks via subgraph explorations. In *International Conference on Machine Learning*, 2021. 7

Zemel, R., Wu, Y., Swersky, K., Pitassi, T., and Dwork, C. Learning fair representations. In *International Conference on Machine Learning*, 2013. 10

Zhang, M. and Chen, Y. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems*, 2018. 1

Zou, D., Balan, R., and Singh, M. On Lipschitz bounds of general convolutional neural networks. *IEEE Transactions on Information Theory*, 66(3):1738–1759, 2019. 7

A. Related Works

Lipschitz Bounds in Deep Models. Prior research on Lipschitz constants has primarily focused on specific types of neural networks incorporating convolutional or attention layers (Zou et al., 2019; Terris et al., 2020; Kim et al., 2021; Araujo et al., 2021). In the context of GNNs, (Dasoulas et al., 2021) introduced a Lipschitz normalization method for self-attention layers in GATs. More recently, (Gama & Sojoudi, 2022) estimated the filter Lipschitz constant using the infinite norm of a matrix. In contrast, the Lipschitz matrix in our study follows a distinct definition and employs different choices of norm types. Additionally, our objective is to enhance the stability of GNNs against unfair biases, which is not clearly addressed in the aforementioned works.

Fair Graph Learning. Fair graph learning is a relatively open domain (Wu et al., 2021; Dai & Wang, 2021; Buyl & De Bie, 2020). Some existing approaches address fairness concerns through techniques such as fairness-aware augmentations or adversarial training. For instance, Fairwalk (Rahman et al., 2019b) is a random walk-based algorithm that aims to mitigate fairness issues in graph node embeddings. Adversarial training is employed in approaches like Compositional Fairness (Bose & Hamilton, 2019) to disentangle learned embeddings from sensitive features. Information Regularization (Liao et al., 2021) utilizes adversarial training to minimize the marginal difference between vertex representations. In addition, (Palowitch & Perozzi, 2020) improves group fairness by ensuring that node embeddings lie on a hyperplane orthogonal to sensitive features. However, there remains ample room for further exploration in rank-based individual fairness (Dong et al., 2021), which is the focus of our work.

Understanding Learning on Graphs. Various approaches have emerged to understand the underlying patterns in graph data and its components. Explanatory models for learning on graphs (Ying et al., 2019; Huang et al., 2022; Yuan et al., 2021; Chen et al., 2023) provide insights into the relationship between a model's predictions and elements in graphs. These works shed light on how local elements or node characteristics influence the decision-making process of GNNs. However, our work differs in that we investigate the impact of Lipschitz constants on practical training dynamics, rather than focusing on trained/fixed-parameter model inference or the influence of local features on GNN decision-making processes.

B. Preliminaries

Lipschitz Constant Let us start by reviewing some definitions related to Lipschitz functions. A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is established to be Lipschitz continuous on an input set $\mathcal{X} \subseteq \mathbb{R}^n$ if there exists a constant $K \geq 0$ such that for all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$, f satisfies the following inequality:

$$\|f(\boldsymbol{x}) - f(\boldsymbol{y})\| \leq K \|\boldsymbol{x} - \boldsymbol{y}\|, \forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}. \tag{6}$$

The smallest possible K in Equation (6) is the Lipschitz constant of f, denoted as $\mathrm{Lip}(f)$:

$$\mathrm{Lip}(f) = \sup_{\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}, \boldsymbol{x} \neq \boldsymbol{y}} \frac{\|f(\boldsymbol{x}) - f(\boldsymbol{y})\|}{\|\boldsymbol{x} - \boldsymbol{y}\|}, \tag{7}$$

and we say that f is a K-Lipschitz function. *The Lipschitz constant of a function is essentially the largest possible change of the output corresponding to a perturbation of the input of unit norm.* This makes the Lipschitz constant of a neural network an important measure of its stability with respect to the input features. However, finding the exact constant can be challenging, so obtaining an upper bound is often the approach taken. Such an upper bound is called a Lipschitz bound.

Graph Neural Networks We assume a given graph $G = G(V, E)$, where V denotes the set of nodes and E denotes the set of edges. We will use $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N\} \subset \mathbb{R}^F$ to denote the N node features in $\mathbb{R}^F$, as the input of any layer of a GNN. By abuse of notation, when there is no confusion, we also follow GNN literature and consider $\boldsymbol{X}$ as the $\mathbb{R}^{N \times F}$ matrix whose i-th row is given by $\boldsymbol{x}_i^\top$, $i = 1, \cdots, N$, though it unnecessarily imposes an ordering of the graph nodes.

GNNs are functions that operate on the adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ of a graph G. Specifically, an L-layer GNN can be defined as a function $f : \mathbb{R}^{N \times F^{\text{in}}} \to \mathbb{R}^{N \times F^{\text{out}}}$ that depends on $\boldsymbol{A}$. Formally, we adopt the following definition:

Definition 1. *An L-layer GNN is a function f that can be expressed as a composition of L message-passing layers h^l and $L-1$ activation functions ρ^l, as follows:*

$$f = h^L \circ \rho^{L-1} \circ \cdots \circ \rho^1 \circ h^1, \tag{8}$$

where $h^l : \mathbb{R}^{F^{l-1}} \to \mathbb{R}^{F^l}$ is the l-th message-passing layer, $\rho^l : \mathbb{R}^{F^l} \to \mathbb{R}^{F^l}$ is the non-linear activation function in the l-th layer, and F^{l-1} and F^l denote the input and output feature dimensions for the l-th message-passing layer h^l, respectively. In addition, we set $l = 1, \cdots, L$.

Rank-based Individual Fairness of GNNs Rank-based Individual Fairness on GNNs (Dong et al., 2021) focuses on the relative ordering of instances rather than their absolute predictions. It ensures that similar instances, as measured by the similarity measure $S(\cdot, \cdot)$, receive consistent rankings or predictions. The criterion requires that if instance i is more similar to instance j than to instance k, then the predicted ranking of j should be higher than that of k, *consistently*. This can be expressed as if $S(\boldsymbol{x}_i, \boldsymbol{x}_j) > S(\boldsymbol{x}_i, \boldsymbol{x}_k)$, then $\boldsymbol{Y}_{ij} > \boldsymbol{Y}_{ik}$, where $\boldsymbol{Y}_{ij}$ and $\boldsymbol{Y}_{ik}$ denote the predicted rankings or predictions for instances $\boldsymbol{x}_j$ and $\boldsymbol{x}_k$, respectively, based on the input instance $\boldsymbol{x}_i$. The criterion ensures that the predicted rankings or predictions align with the relative similarities between instances, promoting fairness and preventing discriminatory predictions based on irrelevant factors.

C. Proofs

C.1. Notations

We use the following notations throughout the paper. Sets are denoted by $\{\}$ and vectors by $()$. For $n \in \mathbb{N}$, we denote $[n] = \{1, \cdots, n\}$. Scalars are denoted by regular letters, lowercase bold letters denote vectors, and uppercase bold letters denote matrices. For instance, $\boldsymbol{x} = (x_1, \cdots, x_n)^\top \in \mathbb{R}^n$ and $\boldsymbol{X} = [X_{ik}]_{i\in[n],k\in[m]} \in \mathbb{R}^{n\times m}$. For any vector $\boldsymbol{x} \in \mathbb{R}^n$, we use $\|\boldsymbol{x}\|$ to denote its ℓ_2-norm: $\|\boldsymbol{x}\| = \left(\sum_{i=1}^n x_i^2\right)^{1/2}$. For any matrix $\boldsymbol{X} \in \mathbb{R}^{n\times m}$, we use $\boldsymbol{X}_{i,:}$ to denote its i-th row and $\boldsymbol{X}_{:,k}$ to denote its k-th column. The $(\infty, 2)$-norm of $\boldsymbol{X}$ is denoted by $\|\boldsymbol{X}\|_{\infty,2} = \max_{i\in[n]} \|\boldsymbol{X}_{i,:}\|$. Given a graph $G = G(V, E)$ with ordered nodes, we denote its adjacency matrix by $\boldsymbol{A}$ such that $A_{ij} = 1$ if $\{i, j\} \in E$ and $A_{ij} = 0$ otherwise. When it is clear from the context, we use $\boldsymbol{X} \in \mathbb{R}^{N\times F}$ to denote a feature matrix whose i-th row corresponds to the features of the i-th node, and the j-th column represents the features across all nodes for the j-th attribute. We denote the output of the GNN as $\boldsymbol{Y} \in \mathbb{R}^{N\times C}$, where N is the number of nodes and C is the number of output classes.

C.2. Proof of Lemma 1 in Section 2.1

Lemma 1. *For any vectors x and y in a Euclidean space, let $g(x)$ and $g(y)$ be vector-valued functions of x and y respectively, and let g_i be the i-th component function of g. Then, we have the following inequality:*

$$\frac{\|g(x) - g(y)\|}{\|x - y\|} \leqslant \left\| \left[\frac{g_i(x) - g_i(y)}{\|x - y\|} \right]_{i=1}^n \right\|, \tag{9}$$

Lemma 1 provides an inequality that relates the norm of the difference between two vector-valued functions, $g(x)$ and $g(y)$, to the norm of a vector composed of the component-wise differences of the functions evaluated at x and y. Based on Lemma 1, we can now present the following theorem:

Proof. We begin by observing that

$$\begin{aligned} \frac{\|g(\boldsymbol{x}) - g(\boldsymbol{y})\|}{\|\boldsymbol{x} - \boldsymbol{y}\|} &= \frac{\|[g_i(\boldsymbol{x}) - g_i(\boldsymbol{y})]_{i=1}^n\|}{\|\boldsymbol{x} - \boldsymbol{y}\|} \\ &= \left\| \left[\frac{|g_i(\boldsymbol{x}) - g_i(\boldsymbol{y})|}{\|\boldsymbol{x} - \boldsymbol{y}\|} \right]_{i=1}^n \right\|. \end{aligned} \tag{10}$$

Furthermore, for each $i \in [n]$, $\dfrac{|g_i(\boldsymbol{x}) - g_i(\boldsymbol{y})|}{\|\boldsymbol{x} - \boldsymbol{y}\|} \le \mathrm{Lip}(g_i)$. Therefore, we can write

$$\begin{aligned} \mathrm{Lip}(g) &= \sup_{\boldsymbol{x}\neq\boldsymbol{y}} \frac{\|g(\boldsymbol{x}) - g(\boldsymbol{y})\|}{\|\boldsymbol{x} - \boldsymbol{y}\|} \\ &= \sup_{\boldsymbol{x}\neq\boldsymbol{y}} \left\| \left[\frac{|g_i(\boldsymbol{x}) - g_i(\boldsymbol{y})|}{\|\boldsymbol{x} - \boldsymbol{y}\|} \right]_{i=1}^n \right\| \\ &\le \sup_{\boldsymbol{x}\neq\boldsymbol{y}} \|[\mathrm{Lip}(g_i)]_{i=1}^n\| = \|[\mathrm{Lip}(g_i)]_{i=1}^n\|, \end{aligned} \tag{11}$$

this completes the proof. □

In the above proof of Lemma 1, we start by rewriting the norm of the difference between $g(\boldsymbol{x})$ and $g(\boldsymbol{y})$ divided by the norm of $\boldsymbol{x} - \boldsymbol{y}$ as a norm of a vector containing the component-wise differences of $g_i(\boldsymbol{x})$ and $g_i(\boldsymbol{y})$ divided by the norm of $\boldsymbol{x} - \boldsymbol{y}$ for each i. We then observe that for each i, the absolute value of $g_i(\boldsymbol{x}) - g_i(\boldsymbol{y})$ divided by the norm of $\boldsymbol{x} - \boldsymbol{y}$ is bounded by the Lipschitz constant $\mathrm{Lip}(g_i)$. Hence, the Lipschitz constant of g is bounded by the norm of the vector $[\mathrm{Lip}(g_i)]_{i=1}^n$. This establishes the inequality in Lemma 1.

C.3. Proof of Theorem 1 in Section 2.1

Theorem 1. *Let $\boldsymbol{Y}$ be the output of an L-layer GNN (represented in $f(\cdot)$) with $\boldsymbol{X}$ as the input. Assuming the activation function (represented in $\rho(\cdot)$) is ReLU with a Lipschitz constant of $\mathrm{Lip}(\rho) = 1$, then the global Lipschitz constant of the GNN, denoted as $\mathrm{Lip}(f)$, satisfies the following inequality:*

$$\mathrm{Lip}(f) \leqslant \max_j \prod_{l=1}^{L} \left\| F^{l'} \right\| \left\| \left[\mathcal{J}(h^l) \right]_j \right\|_\infty, \tag{12}$$

where $F^{l'}$ represents the output dimension of the l-th message-passing layer, j is the index of the node (e.g., j-th), and the vector $\left[\mathcal{J}(h^l)\right] = \left[\left\|\boldsymbol{J}_1(h^l)\right\|, \left\|\boldsymbol{J}_2(h^l)\right\|, \cdots, \left\|\boldsymbol{J}_{F^{l'}}(h^l)\right\|\right]$. Notably, $\boldsymbol{J}_i(h^l)$ denotes the i-th row of the Jacobian matrix of the l-th layer's input and output, and $\left[\mathcal{J}(h^l)\right]_j$ is the vector corresponding to the j-th node in the l-th layer $h^l(\cdot)$.

Theorem 1 provides an inequality that bounds the global Lipschitz constant of the GNN based on the layer outputs and Jacobian matrices. It is derived as follows:

Proof. We begin by examining the Lipschitz property of the GNN. Let $\boldsymbol{Y}$ denote the output of an L-layer GNN with input $\boldsymbol{X}$. Assuming the commonly used ReLU activation function as the non-linear layer $\rho(\cdot)$, we have $\mathrm{Lip}(\rho) = 1$. First, we consider the Lipschitz constant between the hidden states of two nodes output by any message-passing layer $h(\cdot)$ in $f(\cdot)$. Let z_1 and z_2 represent the hidden states of node features x_1 and x_2, respectively. The Lipschitz constant between these hidden states is given by:

$$\frac{\|z_1 - z_2\|}{\|x_1 - x_2\|} = \frac{\| \left(h(x_1) - h(x_2)\right) \|}{\|x_1 - x_2\|}. \tag{13}$$

By applying the triangle inequality, we obtain:

$$\frac{\|z_1 - z_2\|}{\|x_1 - x_2\|} \leqslant \left\| \left[\frac{h(x_1)_i - h(x_2)_i}{\|x_1 - x_2\|} \right]_{i=1}^{F'} \right\|, \tag{14}$$

Next, we consider the Lipschitz constant between individual elements of the hidden states. Let $f(x_1)$ and $f(x_2)$ represent the hidden state matrices for inputs x_1 and x_2, respectively. By again applying the triangle inequality, we have:

$$\frac{\|z_1 - z_2\|}{\|x_1 - x_2\|} \leqslant \left\| F' \times \max_i \frac{h(x_1)_i - h(x_2)_i}{\|x_1 - x_2\|} \right\|, \tag{15}$$

let's focus on the Lipschitz constant of the individual elements, $\frac{h(x_1)_i - h(x_2)_i}{\|x_1 - x_2\|}$: Here, $f(x)_i$ denotes the i-th column of the matrix $f(x)$. We denote $h^l(\cdot)$ as the operation of the l-th message-passing layer in $f(\cdot)$, then by applying the triangle inequality and leveraging the Lipschitz property of the ReLU activation function, we have:

$$\frac{\|f(x_1)_{j,1} - f(x_2)_{j,2}\|}{\|x_{j,1} - x_{j,2}\|} \leqslant \prod_{l=1}^{L} \left\| F^{l'} \right\| \left\| \left[\mathcal{J}(h^l) \right]_j \right\|_\infty, \tag{16}$$

where $x_{j,1}$ and $x_{j,1}$ denote features of j-th node's in x_1 and x_2, respectively, and $\left[\mathcal{J}(h^l)\right]_j$ represents the j-th node's the Jacobian matrix of the l-th message-passing layer. Therefore, the Lipschitz constant for the GNN can be expressed as:

$$\mathrm{Lip}(f) = \max_j \prod_{l=1}^{L} \left\| F^{l'} \right\| \left\| \left[\mathcal{J}(h^l) \right]_j \right\|_\infty. \tag{17}$$

In summary, we have shown that for any two input samples x_1 and x_2, the Lipschitz constant of the GNN, denoted as $\mathrm{Lip}(f)$, satisfies:

$$\|\boldsymbol{Y}_1 - \boldsymbol{Y}_2\| \leqslant \mathrm{Lip}(f) \, \|\boldsymbol{X}_1 - \boldsymbol{X}_2\|, \tag{18}$$

where $\boldsymbol{Y}$ denotes the output of the GNN for inputs $\boldsymbol{X}$. This inequality implies that the Lipschitz constant $\mathrm{Lip}(f)$ controls the magnitude of changes in the output based on input biases/perturbations. Therefore, we have established the following result:

$$\|\boldsymbol{Y}_1 - \boldsymbol{Y}_2\| \leqslant \prod_{l=1}^{L} \left\| F^{l'} \right\| \left\| \left[\mathcal{J}(h^l) \right]_j \right\|_\infty \|\boldsymbol{X}_1 - \boldsymbol{X}_2\|. \tag{19}$$

This inequality demonstrates that the Lipschitz constant of the GNN, $\mathrm{Lip}(f)$, controls the magnitude of the difference in the output $\boldsymbol{Y}$ based on the difference in the input $\boldsymbol{X}$. It allows us to analyze the stability of the model's output with respect to input perturbations. □

D. Model Card

D.1. Implementations

Code and datasets will be publicly available. Algorithm 1 is a PyTorch-style Pseudocode:

Algorithm 1 JacoLip: A simplified PyTorch-style Pseudocode of our Lipschitz Bounds for fairness.

```
# model: graph neural network model
# Train model for N epochs
for X, A, target in dataloader_mlp:
  pred = model(X, A)
  ce_loss = CrossEntropyLoss(pred, target)

  # Compute Lipschitz constant for input
  jacobian = Jaco(X, target)
  model_lip = Lip(jacobian) # Eq.(8)
  global_lip = norm(model_lip) # Eq.(9)

  # Optimize model with Lipschitz bound
  loss = ce_loss + u * global_lip
  loss.backward()
  optimizer.step()
```

D.2. Hyperparameters Configurations

The hyper-parameters for our method across all datasets are listed in Table 2. For fair comparisons, we follow the default settings of Redress (Dong et al., 2021).

E. Detailed Setup

Datasets We evaluate the effectiveness of the Lipschitz bound in promoting individual fairness in GNNs from a ranking perspective by conducting experiments on three real-world datasets, each for a chosen downstream task (node classification or link prediction). Specifically, we use one citation network (`ACM` (Tang et al., 2008)) and two co-authorship networks (`Co-author-CS` and `Co-author-Phy` (Shchur et al., 2018) from the KDD Cup 2016 challenge) for the node classification task. For the link prediction task, we use three social networks (`BlogCatalog` (Tang & Liu, 2009), `Flickr` (Huang et al., 2017), and `Facebook` (Leskovec & Mcauley, 2012). We follow their public train/val/test splits provided by a prior

Table 2: Hyperparameters used in our experiments.

Hyperparameters	Node Classification			Link Prediction		
	ACM	Coauthor-CS	Coauthor-Phy	Blog	Flickr	Facebook
Hyperparameters w.r.t. the GCN model						
# Layers	2	2	2	2	2	2
Hidden Dimension	[16, 9]	[16, 15]	[16, 5]	[32, 16]	[32, 16]	[32, 16]
Activation	ReLU used for all datasets					
Dropout	0.03	0.03	0.03	0.00	0.00	0.00
Optimizer	AdamW with $1e-5$ weight decay used for all datasets					
Pretrain Steps	300	300	300	200	200	200
Training Steps	150	200	200	60	100	50
Learning Rate	0.01	0.01	0.01	0.01	0.01	0.01
Hyperparameters w.r.t. the SGC model						
# Layers	1	1	1	N.A.	N.A.	N.A.
Hidden Dimension	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.
Dropout	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.
Optimizer	AdamW with $1e-5$ weight decay			N.A.	N.A.	N.A.
Pretrain Steps	300	500	500	N.A.	N.A.	N.A.
Training Steps	15	40	30	N.A.	N.A.	N.A.
Learning Rate	0.01	0.01	0.01	N.A.	N.A.	N.A.
Hyperparameters w.r.t. the GAE model						
# Layers	N.A.	N.A.	N.A.	2	2	2
Hidden Dimension	N.A.	N.A.	N.A.	[32, 16]	[32, 16]	[32, 16]
Activation	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.
Dropout	0.0	0.0	0.0	0.0	0.0	0.0
Optimizer	N.A.	N.A.	N.A.	AdamW with $1e-5$ weight decay		
Pretrain Steps	N.A.	N.A.	N.A.	200	200	200
Training Steps	N.A.	N.A.	N.A.	60	100	50
Learning Rate	N.A.	N.A.	N.A.	0.01	0.01	0.01

rank-based individual fairness work (Dong et al., 2021). The datasets used in our work are referred to as CS and Phy, which are abbreviations for the Co-author-CS and Co-author-Phy datasets, respectively. A comprehensive overview of the datasets, including their detailed statistics, is presented in Table 3.

Backbones We employ two widely-used GNN architectures as backbone models for each downstream learning task in our experiments. Specifically, for the node classification task, we adopt Graph Convolutional Network (GCN) (Kipf & Welling, 2017) and Simplifying Graph Convolutional Network (SGC) (Wu et al., 2019). For the link prediction task, we use GCN and Variational Graph Auto-Encoders (GAE) (Kipf & Welling, 2016).

Baselines In the previous work on rank-based individual fairness (Dong et al., 2021), existing group fairness graph embedding methods, such as (Bose & Hamilton, 2019; Rahman et al., 2019a), are unsuitable for comparison as they promote fairness for subgroups determined by specific protected attributes, whereas our focus is on individual fairness without such attributes. To evaluate our proposed method against this notion of individual fairness, we compare it with three important baselines for rank-based individual fairness:

- *Redress* (Dong et al., 2021): This method proposes a rank-based framework to enhance the individual fairness of GNNs. It integrates GNN model utility maximization and rank-based individual fairness promotion in a joint framework to enable end-to-end training.
- *InFoRM* (Kang et al., 2020): InFoRM is an individual fairness framework for conventional graph mining tasks, such as PageRank and Spectral Clustering, based on the Lipschitz condition. We adapt InFoRM to different GNN backbone models by combining its individual fairness promotion loss and the unity loss of the GNN backbone model, and optimizing the final loss in an end-to-end manner.
- *PFR* (Lahoti et al., 2019b): PFR aims to learn fair representations to achieve individual fairness. It outperforms traditional approaches, such as (Hardt et al., 2016; Zemel et al., 2013; Lahoti et al., 2019a), in terms of individual fairness promotion. As PFR can be considered a preprocessing strategy and is not tailored for graph data, we use it on the input node features to generate a new fair node feature representation.

Evaluation Metrics To provide a comprehensive evaluation of rank-based individual fairness, we use two key metrics: the classification accuracy *Acc.* for the node classification task, and the area under the receiver operating characteristic curve *AUC* for the link prediction task. For the individual fairness evaluation, we use a widely used ranking metric following the previous work (Dong et al., 2021): *NDCG@k* (Järvelin & Kekäläinen, 2002). This metric allows us to measure the similarity between the rankings generated from $\boldsymbol{S_Y}$ (result similarity matrix) and $\boldsymbol{S_G}$ (Oracle similarity matrix) for each node. We report the average values of NDCG@k across all nodes and set $k = 10$ for quantitative performance comparison.

Table 3: Detailed statistics of the datasets used for node classification and link prediction. We follow the default settings of Redress (Dong et al., 2021) fair comparisons.

Task	Dataset	# Nodes	# Edges	# Features	# Classes
node cls.	`ACM`	$16,484$	$71,980$	$8,337$	9
	`Coauthor-CS`	$18,333$	$81,894$	$6,805$	15
	`Coauthor-Phy`	$34,493$	$247,962$	$8,415$	5
link pred.	`BlogCatalog`	$5,196$	$171,743$	$8,189$	N.A.
	`Flickr`	$7,575$	$239,738$	$12,047$	N.A.
	`Facebook`	$4,039$	$88,234$	$1,406$	N.A.

F. Additional Experiments

F.1. Effectiveness on node classification tasks

According to Table 4, on node classification tasks, JacoLip consistently shows a competitive or improved trade-off between accuracy and error compared to the baselines, highlighting the effectiveness of the Lipschitz bound in promoting individual fairness on graphs.

F.2. Effectiveness on link prediction tasks

Similar observations can be made for the link prediction task from Table 5, where the performance of Vanilla (GCN or GAE), InFoRM, PFR, Redress, and JacoLip methods is evaluated using AUC for utility and NDCG@10 for fairness. In both tasks, JacoLip consistently demonstrates competitive or improved performance compared to the baselines, highlighting the effectiveness of the Lipschitz bound in promoting individual fairness on graphs.

F.3. Impact of Lipschitz Bounds on Training Dynamics

We further analyze the impact of Lipschitz bounds through the optimization process and explore the its interactions with weight parameters, gradient, fairness, and accuracy during training in Figure 1:

For the nonlinear GCN model, our proposed JacoLip demonstrates positive regularizations on gradients. Particularly, at the initial epochs, JacoLip effectively stabilizes gradient magnitudes, leading to higher accuracy (e.g., $\sim$20% on feature similarity and $\sim$40% on structural similarity) compared to the baseline Redress. Additionally, during these initial epochs, JacoLip maintains higher fairness metrics NDCG (e.g., $\sim$0.15 on feature similarity) compared to the baseline Redress on GCN; *For the linear SGC model*, JacoLip also achieves a favorable trade-off between fairness and accuracy compared to the baseline Redress approach. At the start of training, JacoLip on SGC exhibits better fairness (e.g., $\sim$0.2 NDCG on feature similarity) than the baseline Redress on SGC, with only a slight decrease in accuracy. Furthermore, as the number of epochs increases, the accuracy of JacoLip on SGC tends to converge to or even surpass the baseline Redress on SGC.

In summary, the observed accuracy-fairness trade-off during training can be attributed to the constraint effect of Lipschitz bounds on gradient optimization: The higher expressivity of the nonlinear GCN makes it more prone to losing consistency in the input-output similarity rank, which is crucial for fairness. In contrast, the simplicity of the linear model preserves consistency more easily, and our JacoLip prioritizes accuracy in this case. These findings offer valuable insights into the dynamic behavior of model training under Lipschitz bounds and highlight the advantages of JacoLip in promoting fairness while maintaining competitive accuracy.

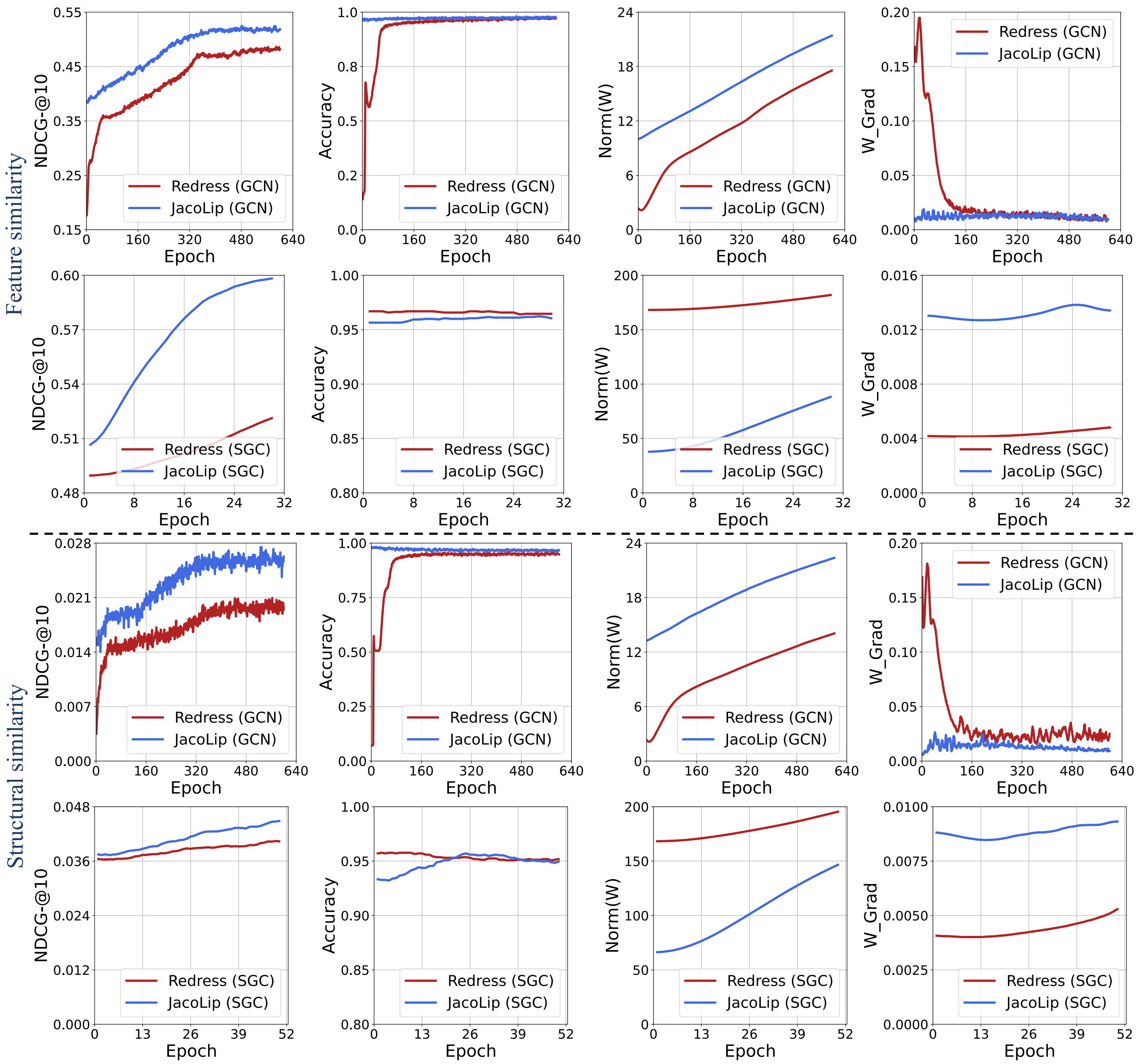

Figure 1: Study of the Lipschitz bounds' impact on model training for rank-based individual fairness. We perform experiments on the co-author-Physics dataset using both nonlinear (GCN) and linear (SGC) models. The training dynamics are assessed by monitoring the NDCG, accuracy, weight norm, and weight gradient as the number of epochs increases. *Upper two rows*: Metrics under feature similarity; *Lower two rows*: Metrics under structural similarity.

Table 4: Evaluation on node classification tasks: comparing under accuracy and error.

Data	Model	Fair Alg.	Feature Similarity		Structural Similarity	
			utility: Acc.↑	fairness: Err.@10↑	utility: Acc.↑	fairness: Err.@10↑
ACM	GCN	Vanilla (Kipf & Welling, 2017)	72.49±0.6	75.70±0.6	72.49±0.6	37.55±0.4
		InFoRM (Kang et al., 2020)	67.65±1.0(−6.68%)	73.49±0.5(−2.92%)	65.91±0.2(−9.07%)	19.96±0.6(−46.8%)
		PFR (Lahoti et al., 2019b)	68.48±0.6(−5.53%)	76.28±0.1(0.77%)	70.22±0.7(−3.13%)	36.54±0.4(−2.69%)
		Redress (Dong et al., 2021)	73.46±0.2(+1.34%)	82.27±0.1(+8.68%)	71.87±0.4(−0.86%)	43.74±0.0(+16.5%)
		JacoLip (on Vanilla)	72.80±0.2(+4.27%)	82.88±0.1(+9.48%)	72.30±0.4(−0.26%)	39.28±0.2(+4.61%)
		JacoLip (on Redress)	71.05±0.4(+2.00%)	82.21±0.3(+8.60%)	71.92±0.3(−0.79%)	46.13±0.3(+22.85%)
	SGC	Vanilla (Wu et al., 2019)	68.40±1.0	80.06±0.1	68.40±1.0	45.95±0.3
		InFoRM (Kang et al., 2020)	67.96±0.5(−0.64%)	75.63±0.5(−5.53%)	66.16±0.6(−3.27%)	39.79±0.1(−13.4%)
		PFR (Lahoti et al., 2019b)	67.69±0.4(−1.04%)	76.80±0.1(−4.07%)	66.69±0.3(−2.50%)	46.99±0.5(+2.26%)
		Redress (Dong et al., 2021)	66.51±0.3(−2.76%)	82.32±0.3(+2.82%)	67.10±0.7(−1.90%)	49.02±0.2(+4.76%)
		JacoLip (on Vanilla)	74.04±0.2(+8.25%)	82.73±0.7(+5.18%)	72.91±0.9(+3.33%)	48.64±0.2(+5.85%)
		JacoLip (on Redress)	69.91±0.1(+2.21%)	85.22±0.4(+6.45%)	71.27±0.3(+4.20%)	52.0±0.4(+13.23%)
CS	GCN	Vanilla (Kipf & Welling, 2017)	90.59±0.3	80.41±0.1	90.59±0.3	26.69±1.3
		InFoRM (Kang et al., 2020)	88.37±0.9(−2.45%)	80.63±0.6(+0.27%)	87.10±0.9(−3.85%)	29.68±0.6(+11.2%)
		PFR (Lahoti et al., 2019b)	87.62±0.2(−3.28%)	76.26±0.1(−5.16%)	85.66±0.7(−5.44%)	19.80±1.4(−25.8%)
		Redress (Dong et al., 2021)	90.06±0.5(−0.59%)	83.24±0.2(+3.52%)	89.91±0.2(−0.86%)	32.42±1.6(+21.5%)
		JacoLip (on Vanilla)	90.41±0.4(−0.20%)	82.57±0.1(+2.69%)	89.12±0.1(−1.62%)	32.8±0.6(+22.74%)
		JacoLip (on Redress)	90.30±0.3(−0.32%)	88.11±0.3(+9.58%)	89.93±0.2(−0.73%)	42.5±0.4(+59.24%)
	SGC	Vanilla (Wu et al., 2019)	87.48±0.8	90.58±0.1	87.48±0.8	43.28±0.2
		InFoRM (Kang et al., 2020)	87.31±0.5(−0.19%)	90.64±0.1(+0.07%)	88.21±0.4(+0.83%)	44.37±0.1(+0.21%)
		PFR (Lahoti et al., 2019b)	87.95±0.2(+0.54%)	79.85±0.2(−11.8%)	86.93±0.1(−0.63%)	38.83±0.8(−10.3%)
		Redress (Dong et al., 2021)	90.48±0.2(+3.43%)	92.03±0.1(+1.60%)	90.39±0.1(+3.33%)	45.81±0.0(+5.85%)
		JacoLip (on Vanilla)	90.71±0.3(+3.69%)	90.75±0.4(+0.19%)	90.34±1.0(+3.27%)	43.92±0.3(+1.48%)
		JacoLip (on Redress)	92.22±0.2(+5.42%)	92.22±0.4(+1.81%)	90.54±0.3(+3.50%)	46.39±0.5(+7.19%)
Phy	GCN	Vanilla (Kipf & Welling, 2017)	94.81±0.2	73.25±0.3	94.81±0.2	2.58±0.1
		InFoRM (Kang et al., 2020)	88.67±0.7(−6.48%)	73.80±0.6(+0.75%)	94.68±0.2(−0.14%)	2.45±0.1(−5.04%)
		PFR (Lahoti et al., 2019b)	88.79±0.2(−6.35%)	73.22±0.4(+0.10%)	89.69±1.0(−5.40%)	1.67±0.1(−35.3%)
		Redress (Dong et al., 2021)	93.71±0.1(−1.16%)	80.23±0.1(+9.53%)	93.91±0.4(−0.95%)	3.22±0.3(+22.9%)
		JacoLip (on Vanilla)	93.71±0.2(−1.16%)	78.64±1.1(+7.36%)	94.75±0.3(−0.06%)	2.75±0.6(+6.80%)
		JacoLip (on Redress)	93.79±0.8(−1.08%)	82.6±0.3(+12.70%)	93.98±0.3(−0.88%)	4.0±0.1(+55.43%)
	SGC	Vanilla (Wu et al., 2019)	94.45±0.2	77.48±0.2	94.45±0.2	4.50±0.1
		InFoRM (Kang et al., 2020)	92.06±0.2(−2.53%)	75.13±0.4(−3.03%)	94.27±0.1(−0.19%)	4.44±0.0(−1.33%)
		PFR (Lahoti et al., 2019b)	87.39±1.2(−7.47%)	73.42±0.2(−5.24%)	89.16±0.3(−5.60%)	3.41±0.2(−24.2%)
		Redress (Dong et al., 2021)	94.81±0.2(+0.38%)	79.57±0.2(+2.70%)	94.54±0.1(+0.10%)	4.98±0.1(+10.7%)
		JacoLip (on Vanilla)	94.43±0.7(−0.02%)	78.82±0.8(+1.73%)	94.09±0.6(−0.38%)	4.75±0.2(+5.56%)
		JacoLip (on Redress)	94.78±0.1(+0.35%)	82.21±0.2(+6.10%)	93.00±1.3(−1.54%)	5.45±0.1(+1.90%)

Table 5: Evaluation on link prediction tasks: comparing under AUC and NDCG.

Data	Model	Fair Alg.	Feature Similarity		Structural Similarity	
			utility: AUC↑	fairness: NDCG@10↑	utility: AUC↑	fairness: NDCG@10↑
Blog	GCN	Vanilla (Kipf & Welling, 2017)	85.87±0.1	16.73±0.1	85.87±0.1	32.47±0.5
		InFoRM (Kang et al., 2020)	79.85±0.6(−7.01%)	15.57±0.2(−6.93%)	84.00±0.1(−2.18%)	26.18±0.3(−19.4%)
		PFR (Lahoti et al., 2019b)	84.25±0.2(−1.89%)	16.37±0.0(−2.15%)	83.88±0.0(−2.32%)	29.60±0.4(−8.84%)
		Redress (Dong et al., 2021)	86.49±0.8(+0.72%)	17.66±0.2(+5.56%)	86.25±0.3(+0.44%)	34.62±0.7(+6.62%)
		JacoLip (on Vanilla)	86.51±0.2(+0.74%)	17.70±0.6(+5.79%)	86.90±0.5(+1.67%)	35.00±0.4(+7.79%)
		JacoLip (on Redress)	85.91±0.2(+0.04%)	18.02±0.6(+7.71%)	86.84±0.5(+1.13%)	35.85±0.4(+10.4%)
	GAE	Vanilla (Kipf & Welling, 2016)	85.72±0.1	17.13±0.1	85.72±0.1	41.99±0.4
		InFoRM (Kang et al., 2020)	80.01±0.2(−6.66%)	16.12±0.2(−5.90%)	82.86±0.0(−3.34%)	27.29±0.3(−35.0%)
		PFR (Lahoti et al., 2019b)	83.83±0.1(−2.20%)	16.64±0.0(−2.86%)	83.87±0.1(−2.16%)	35.91±0.4(−14.5%)
		Redress (Dong et al., 2021)	84.67±0.9(−1.22%)	18.19±0.1(+6.19%)	86.36±1.5(+0.75%)	43.51±0.7(+3.62%)
		JacoLip (on Vanilla)	85.75±0.4(+0.03%)	17.96±0.5(+4.85%)	85.86±0.5(+0.16%)	42.20±0.3(+0.50%)
		JacoLip (on Redress)	85.70±0.4(−0.02%)	18.34±0.5(+7.06%)	86.31±0.5(+0.69%)	43.60±0.3(+3.83%)
Flickr	GCN	Vanilla (Kipf & Welling, 2016)	92.20±0.3	13.10±0.2	92.20±0.3	22.35±0.3
		InFoRM (Kang et al., 2020)	91.39±0.0(−0.88%)	11.95±0.1(−8.78%)	91.73±0.1(−0.51%)	23.28±0.6(+4.16%)
		PFR (Lahoti et al., 2019b)	91.91±0.1(−0.31%)	12.94±0.0(−1.22%)	91.86±0.2(−0.37%)	19.80±0.4(−11.4%)
		Redress (Dong et al., 2021)	91.38±0.1(−0.89%)	13.58±0.3(+3.66%)	92.67±0.2(+0.51%)	28.45±0.5(+27.3%)
		JacoLip (on Vanilla)	92.75±0.3(+0.59%)	13.74±0.4(+4.89%)	92.54±0.1(+0.37%)	26.61±0.4(+19.1%)
		JacoLip (on Redress)	92.53±0.3(+0.35%)	14.37±0.4(+9.69%)	92.69±0.1(+0.53%)	28.65±0.4(+28.2%)
	GAE	Vanilla (Kipf & Welling, 2016)	89.98±0.1	12.77±0.0	89.98±0.1	23.58±0.2
		InFoRM (Kang et al., 2020)	88.76±0.7(−1.36%)	12.07±0.1(−5.48%)	91.51±0.2(+1.70%)	15.78±0.3(−33.1%)
		PFR (Lahoti et al., 2019b)	90.30±0.1(+0.36%)	12.12±0.1(−5.09%)	90.10±0.1(+1.33%)	20.46±0.3(−13.2%)
		Redress (Dong et al., 2021)	89.45±0.5(−0.59%)	14.24±0.1(+11.5%)	89.52±0.3(−0.51%)	29.83±0.2(+26.5%)
		JacoLip (on Vanilla)	89.88±0.3(−0.11%)	14.37±0.1(+12.53%)	89.95±0.2(−0.03%)	28.74±0.5(+21.9%)
		JacoLip (on Redress)	89.92±0.3(−0.06%)	14.85±0.1(+16.29%)	89.56±0.2(−0.46%)	30.04±0.5(+28.7%)
FB	GCN	Vanilla (Kipf & Welling, 2017)	95.60±1.7	23.07±0.2	95.60±1.7	16.55±1.1
		InFoRM (Kang et al., 2020)	90.26±0.1(−5.59%)	23.23±0.3(+0.69%)	96.66±0.6(+1.11%)	15.18±0.7(−8.28%)
		PFR (Lahoti et al., 2019b)	87.11±1.2(−8.88%)	21.83±0.2(−5.37%)	94.87±1.9(−0.76%)	19.53±0.5(+18.0%)
		Redress (Dong et al., 2021)	96.49±1.6(+0.93%)	29.60±0.1(+28.3%)	92.66±0.4(−3.08%)	27.73±1.1(+67.5%)
		JacoLip (on Vanilla)	96.21±0.2(+0.63%)	29.47±0.3(+27.7%)	95.46±0.9(−0.14%)	26.60±0.1(+60.7%)
		JacoLip (on Redress)	96.11±0.2(+0.53%)	30.07±0.3(+30.3%)	92.76±0.9(−2.97%)	28.64±0.1(+73.1%)
	GAE	Vanilla (Kipf & Welling, 2016)	98.54±0.0	26.75±0.1	98.54±0.0	27.03±0.1
		InFoRM (Kang et al., 2020)	90.50±0.4(−8.16%)	22.77±0.2(−14.9%)	95.03±0.1(−3.56%)	15.38±0.2(−43.1%)
		PFR (Lahoti et al., 2019b)	96.91±0.1(−1.65%)	23.52±0.1(−12.1%)	98.28±0.0(−0.26%)	22.89±0.3(−15.3%)
		Redress (Dong et al., 2021)	95.98±1.5(−2.60%)	28.43±0.3(+6.28%)	94.07±1.7(−4.54%)	33.53±0.2(+24.0%)
		JacoLip (on Vanilla)	97.40±0.1(−1.16%)	27.44±0.6(+2.58%)	97.02±1.1(−1.54%)	30.90±0.5(+14.3%)
		JacoLip (on Redress)	96.10±0.1(−2.48%)	28.46±0.6(+6.39%)	94.22±1.1(−4.38%)	31.62±0.5(+17.1%)