PyGDebias: A Python Library for Debiasing in Graph Learning

Yushun Dong
The University of Virginia
Charlottesville, USA
yd6eb@virginia.edu

Zhenyu Lei
The University of Virginia
Charlottesville, USA
vjd5zr@virginia.edu

Zaiyi Zheng
The University of Virginia
Charlottesville, USA
sjc4fq@virginia.edu

Song Wang
The University of Virginia
Charlottesville, USA
sw3wv@virginia.edu

Jing Ma
Case Western Reserve University
Cleveland, USA
jing.ma5@case.edu

Alex Jing Huang
The University of Virginia
Charlottesville, USA
bdt2jr@virginia.edu

Chen Chen
The University of Virginia
Charlottesville, USA
zrh6du@virginia.edu

Jundong Li
The University of Virginia
Charlottesville, USA
jundong@virginia.edu

ABSTRACT

Graph-structured data is ubiquitous among a plethora of real-world applications. However, as graph learning algorithms have been increasingly deployed to help decision-making, there has been rising societal concern in the bias these algorithms may exhibit. In certain high-stake decision-making scenarios, the decisions made may be life-changing for the involved individuals. Accordingly, abundant explorations have been made to mitigate the bias for graph learning algorithms in recent years. However, there still lacks a library to collectively consolidate existing debiasing techniques and help practitioners to easily perform bias mitigation for graph learning algorithms. In this paper, we present PyGDebias, an open-source Python library for bias mitigation in graph learning algorithms. As the first comprehensive library of its kind, PyGDebias covers 13 popular debiasing methods under common fairness notions together with 26 commonly used graph datasets. In addition, PyGDebias also comes with comprehensive performance benchmarks and well-documented API designs for both researchers and practitioners. To foster convenient accessibility, PyGDebias is released under a permissive BSD-license together with performance benchmarks, API documentation, and use examples at https://github.com/yushundong/PyGDebias.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning algorithms**.

KEYWORDS

Algorithmic Bias, Graph Learning, Graph Neural Networks

WWW '24 Companion, May 13–17, 2024, Singapore, Singapore

ACM ISBN 979-8-4007-0172-6/24/05…$15.00
https://doi.org/10.1145/3589335.3651239

ACM Reference Format:
Yushun Dong, Zhenyu Lei, Zaiyi Zheng, Song Wang, Jing Ma, Alex Jing Huang, Chen Chen, and Jundong Li. 2024. PyGDebias: A Python Library for Debiasing in Graph Learning. In *Companion Proceedings of the ACM Web Conference 2024 (WWW '24 Companion), May 13–17, 2024, Singapore, Singapore.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3589335.3651239

1 INTRODUCTION

Graph learning algorithms have been increasingly deployed in a plethora of real-world applications [3, 27]. Nevertheless, there has been a rise in societal concerns about the algorithmic bias these algorithms may exhibit [4, 6, 10]. In certain high-stakes graph learning applications, such as healthcare and criminal justice, life-changing decisions could be made for the involved individuals. Therefore, properly handling the algorithmic bias in commonly used graph learning algorithms has become a critical need.

In recent years, numerous efforts have been made to mitigate the exhibited bias in graph learning. Despite these abundant works, it remains difficult for practitioners to compare existing bias mitigation methods and choose appropriate ones for deployment. We argue that a series of complicated factors have led to such a dilemma, such as experimental setting inconsistencies and evaluation metric differences. In fact, a common way to handle these complex factors is to collect existing methods as a library, which paves the way towards easy implementation, evaluation, and deployment for practitioners. As an example, AIF360 [2] is a library developed to mitigate the bias exhibited by machine learning models over i.i.d. data. Such a library has collected multiple existing debiasing methods and has successfully facilitated the transition of fairness research algorithms to deployment in industrial settings. However, existing libraries mainly only include debiasing methods on regular non-graph data types, and cannot be directly adopted to debias graph learning algorithms due to the unique challenges in graph learning [4, 6, 8, 10, 11]. Therefore, in graph learning, there still lacks a library to collectively consolidate existing debiasing techniques and help practitioners easily mitigate bias in popular algorithms.

To bridge this gap, we design the first comprehensive **P**ython **G**raph **Debias**ing library named PyGDebias, with a series of key

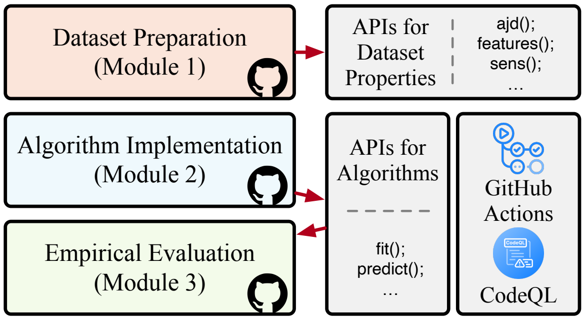

Figure 1: An illustration for the design of PyGDebias.

technical advancements. We summarize the main contributions of PyGDebias in four perspectives below.

- **Diverse Evaluation Metrics.** PyGDebias provides abundant evaluation metrics for model utility and fairness measurements. Notably, the integrated fairness metrics are under a wide range of commonly used fairness notions, including group fairness, individual fairness, and counterfactual fairness, which supports diverse research in this field.
- **Extensive Real-World Datasets.** PyGDebias provides 24 real-world graph-structured datasets in the domain of algorithmic fairness research, including 22 commonly used datasets and two newly collected ones. All datasets are packed up in a standard approach for ease of use, and contributions are also open for further extension.
- **Comprehensive Debiasing Algorithms.** PyGDebias covers a wide range of popular debiasing algorithms in the domain of graph learning, which can be adopted to achieve higher fairness levels under different fairness notions. Specifically, PyGDebias supports 13 popular debiasing algorithms, and is open for further contributions as well.
- **Abundant Materials.** PyGDebias also comes with an extensive performance benchmark under different fairness notions, where experimental settings are ensured to be consistent under each notion. Additionally, all datasets are also integrated with PyGDebias, and the corresponding APIs can be easily called for evaluation.

In addition, to foster convenient accessibility, PyGDebias is released under a permissive BSD-license together with performance benchmarks, API documentation, and use examples at https://github.com/yushundong/PyGDebias.

2 LIBRARY DESIGN

The architecture of PyGDebias encompasses a comprehensive streamline to conduct debiasing for graph learning algorithms. In particular, three different modules are involved in this streamline, namely graph dataset preparation, algorithm implementation, and the subsequent empirical evaluation. We show an overview of the design for PyGDebias in Figure 1, and we provide a brief introduction for each of the three modules below.

Module 1: Dataset Preparation. PyGDebias offers an extensive collection of popular graph datasets, including 22 commonly used datasets in existing literature on graph learning algorithm debiasing, together with 2 newly collected datasets (i.e., AMiner-L and AMiner-S) for further advances of this field. We provide an introduction of all collected datasets in Table 1. Although various raw datasets come in diverse formats and data types, PyGDebias automatically preprocesses all raw datasets to ensure the uniformity, providing a set of datasets in standardized formats and data types. As such, users can easily access these datasets by simply instantiating the designated submodules, which also automatically downloads and stores the raw data from our repository online to user-specified locations. In addition, various properties of each datasets (e.g., the adjacency matrix and the total number of nodes) are easily accessible through unified APIs.

Table 1: 24 commonly used real-world graph datasets collected in PyGDebias 0.1.1.

Dataset	#Nodes	#Edges	#Features
Facebook	1,045	53,498	573
Pokec_z	67,796	882,765	276
Pokec_n	66,569	729,129	265
NBA	403	16,570	95
German	1,000	24,970	27
Credit	30,000	200,526	13
Recidivism	18,876	403,977	18
Google+	290,468	3,601	2,532
AMiner-L	129,726	591,039	5,694
AMiner-S	39,424	52,460	5,694
Cora	2,708	4,751	1,433
Citeseer	3,312	4,194	3,703
Pubmed	19,717	88,648	500
Amazon	2,549 (item) 2 (genre)	2,549	N/A
Yelp	2,834 (item) 2 (genre)	2,834	N/A
Ciao	7,375 (user) 106,797 (product)	57,544	N/A
DBLP	22,166 (user) 296,277 (product)	355,813	N/A
Filmtrust	1,508 (user) 2,071 (item)	35,497	N/A
Lastfm	1,892 (customer) 17,632 (producer)	92,800	N/A
ML100k	943 (user) 1,682 (item)	100,000	4
ML1m	6,040 (user) 3,952 (item)	1,000,209	4
ML20m	138,493 (user) 27,278 (item)	20,000,263	N/A
Oklahoma	3,111	73,230	N/A
UNC	4,018	65,287	N/A

Table 2: 13 debiasing methods collected for graph learning algorithms in PyGDebias 0.1.1.

Methods	Debiasing Technique	Fairness Notions
FairGNN [5]	Adversarial Learning	Group Fairness
EDITS [9]	Edge Rewiring	Group Fairness
FairWalk [22]	Rebalancing	Group Fairness
CrossWalk [17]	Rebalancing	Group Fairness
UGE [28]	Edge Rewiring	Group Fairness
FairVGNN [29]	Adversarial Learning	Group Fairness
FairEdit [18]	Edge Rewiring	Group Fairness
NIFTY [1]	Optimization with Regularization	Group/Counterfactual Fairness
GEAR [19]	Edge Rewiring	Group/Counterfactual Fairness
InFoRM [15]	Optimization with Regularization	Individual Fairness
REDRESS [7]	Optimization with Regularization	Individual Fairness
GUIDE [24]	Optimization with Regularization	Individual Fairness
RawlsGCN [16]	Rebalancing	Degree-Related Fairness

Module 2: Algorithm Implementation. PyGDebias provides a comprehensive coverage of graph debiasing algorithms collected from recent works, including both algorithms based on Graph Neural Networks (GNNs, e.g. FairGNN [5]) and those non-GNN-based (e.g. FairWalk [23]) ones. Initializing and executing the algorithms are straightforward and also highly flexible with unified APIs. We provide an introduction of all collected algorithms in Table 2. In addition, PyGDebias enables customization of different algorithms to meet specific use requirements, which allows for diverse modifications in the structure and hyperparameter configurations. As an example, FairGNN accepts both model-specific configurations (such as backbone GNN model specification) and common training hyperparameters (such as dropout rate and learning rate), which enables model personalization and optimization configuration.
Module 3: Empirical Evaluation. PyDebias is equipped with comprehensive evaluation approaches for the collected algorithms after optimization, and such evaluation can be easily performed on different datasets. Similar to the access of dataset properties, we provide evaluation metrics in standardized formats and these evaluation metrics work by directly taking the output of algorithms as their input. In general, the collected metrics fall into two categories. (1) Utility-based metrics, e.g. Accuracy (`ACC`) and Area Under the Curve (`AUCROC`); (2) Fairness-based metrics, e.g. Demographic Parity (`DP`), Equality Opportunity (`EO`), and Group Disparity of Individual fairness (`GDIF`). An illustrative example of using the model named GUIDE is provided in Demo 1 to show an end-to-end training procedure based on PyGDebias.

In addition, abundant APIs are incorporated in PyGDebias for various functionalities. Below we introduce the API design for (1) dataset properties and (2) algorithm optimization and inference.
APIs for Dataset Properties. We first introduce the API design in PyGDebias to access the properties of the integrated datasets. In general, PyGDebias provide a simple and straightforward design for the ease of use. More specifically, PyGDebias provides eight main APIs to retrieve the corresponding standard properties for most of the algorithms. We provide a brief introduction for each of them below. (1) `adj()` returns the adjacency matrix of a graph, which describes the connectivity between nodes in the dataset. (2) `features()` returns the input feature matrix of all the data samples, e.g. the absence/presence of words for the content of papers in the citation graph dataset. (3) `idx_train(), idx_val(), idx_test()` are the APIs that return the node indices for the training/validation/test sets. (4) `labels()` returns the ground truth labels for the nodes involved in the dataset. (7) `sens()` returns a vector flagging the membership of each node w.r.t. different demographic subgroups (e.g., male v.s. female), which is a particularly interest in group fairness research [10]. It is worth noting that PyGDebias uses a parent class `Dataset` to standardize the dataset pre-processing and access. By simply wrapping a new dataset into a class that inherits from the defined parent class, PyGDebias allows for the easy inclusion of new datasets to the whole streamline.
APIs for Algorithm Optimization & Inference. We then introduce the APIs PyGDebias provided for algorithm optimization and inference. All algorithms (see Table 2) integrated in PyGDebias is packed up as an individual class, and the API design for all algorithms follows a consistent pattern for ease of use. We now

```
from pygdebias.debiasing import GUIDE
from pygdebias.datasets import Nba

# Instantiate the dataset.
nba = Nba()
adj, features, idx_train, idx_val, idx_val, idx_test,
     labels, sens = nba.adj(), nba.features(),
     nba.idx_train(), nba.idx_val(), nba.idx_test(),
     nba.labels(), nba.sens()

# Initiate the model (with default parameters).
model = GUIDE()

# Train the model.
model.fit(adj, features, idx_train, idx_val, idx_test,
     labels, sens)

# Evaluate the model.
(ACC, AUCROC, F1, ACC_sens0, AUCROC_sens0, F1_sens0,
     ACC_sens1, AUCROC_sens1, F1_sens1, SP, EO) =
     model.predict()
```

Demo 1: Using GUIDE [24] on NBA dataset.

introduce two main types of functions that are shared by all integrated algorithms. (1) `fit()` allows users to optimize the model based on either user-specified or default hyperparameters. Different datasets can also serve as the input of this function. (2) `predict()` allows users to perform inference on the test set, and it returns an array of metric values for ease of evaluation. For most of the algorithms, `predict()` returns standardized metrics for evaluation of both model utility and fairness performance. Different metric values under different fairness notions (e.g., group fairness, individual fairness, counterfactual fairness, and degree-related fairness) can also be easily obtained through this function.
Dependencies. PyGDebias is built upon Python 3.6+ with common deep learning packages, including Pandas [20], Numpy [14], Scipy [26], and PyTorch [21] for data processing and model training. Additionally, PyGDebias is also developed based on popular libraries to handle graph-structured data such as NetworkX [13] and Pytorch Geometrics (PyG) [12] frameworks.

3 ROBUSTNESS AND ACCESSIBILITY

Robustness & Quality Check. To facilitate robust automation of our project, we have incorporated automatic code testing with *GitHub Actions*[1]. PyGDebias includes a suite of unit tests for dataset implementations and comprehensive integration tests, which ensures that each update upholds the integrity and stability of the project. Furthermore, PyGDebias has been comprehensively tested with CUDA on local devices. For security assurance, we use *CodeQL*[2] to automate security checks and identify potential vulnerabilities proactively. In adherence to the PEP8 standard [25], our project maintains a uniform black-formatted coding style given by *Pylint*[3], enhancing code readability and community collaborations.
Accessibility. PyGDebias provides clear guidance on installation and dependency requirements in its repository. In addition, a more

[1]GitHub Actions: https://docs.github.com/en/actions.
[2]CodeQL: https://codeql.github.com/docs/index.html
[3]Pylint: https://pylint.pycqa.org/en/latest/index.html

detailed documentation has been published including illustrations about how to utilize each API and execute the whole streamline. The performance benchmark are also provided in the main page of its repository for better comparison and reproduction of existing graph fairness methods. To encourage broader community contributions, PyGDebias is mainly hosted on GitHub, together with a user-friendly guide for contributions and an efficient GitHub-hosted mechanism for reporting issues.

4 CONCLUSION AND FUTURE PLANS

In this paper, we introduced PyGDebias, the first comprehensive library to facilitate research in the realm of algorithmic fairness for graph learning algorithms. PyGDebias supports a variety of fairness algorithms with hands-on APIs, which have been integrated into a standardized workflow. In the future, we will endeavor to achieve several goals: (1) integrate more popular datasets and algorithms; (2) expand its functionality to cover a wider range of real-world graph learning applications; (3) incorporate advanced machine learning techniques such as automatic hyperparameter tuning and curriculum learning to provide a more intelligent streamline; and (4) collaborate with industry stakeholders to enhance utility.

5 ACKNOWLEDGMENTS

This work is supported in part by the National Science Foundation under grants (IIS-2006844, IIS-2144209, IIS-2223769, CNS2154962, and BCS-2228534), the Commonwealth Cyber Initiative Awards under grants (VV-1Q23-007, HV2Q23-003, and VV-1Q24-011), the JP Morgan Chase Faculty Research Award, and the Cisco Faculty Research Award.

REFERENCES

[1] Chirag Agarwal, Himabindu Lakkaraju, and Marinka Zitnik. 2021. Towards a unified framework for fair and stable graph representation learning. In *Uncertainty in Artificial Intelligence*. PMLR, 2114–2124.

[2] Rachel KE Bellamy, Kuntal Dey, Michael Hind, Samuel C Hoffman, Stephanie Houde, Kalapriya Kannan, Pranay Lohia, Jacquelyn Martino, Sameep Mehta, Aleksandra Mojsilović, et al. 2019. AI Fairness 360: An extensible toolkit for detecting and mitigating algorithmic bias. *IBM Journal of Research and Development* 63, 4/5 (2019), 4–1.

[3] Dawei Cheng, Fangzhou Yang, Sheng Xiang, and Jin Liu. 2022. Financial time series forecasting with multi-modality graph neural network. *Pattern Recognition* 121 (2022), 108218.

[4] Manvi Choudhary, Charlotte Laclau, and Christine Largeron. 2022. A survey on fairness for machine learning on graphs. *arXiv preprint arXiv:2205.05396* (2022).

[5] Enyan Dai and Suhang Wang. 2021. Say no to the discrimination: Learning fair graph neural networks with limited sensitive attribute information. In *Proceedings of the 14th International Conference on Web Search and Data Mining*. 680–688.

[6] Enyan Dai, Tianxiang Zhao, Huaisheng Zhu, Junjie Xu, Zhimeng Guo, Hui Liu, Jiliang Tang, and Suhang Wang. 2022. A comprehensive survey on trustworthy graph neural networks: Privacy, robustness, fairness, and explainability. *arXiv preprint arXiv:2204.08570* (2022).

[7] Yushun Dong, Jian Kang, Hanghang Tong, and Jundong Li. 2021. Individual fairness for graph neural networks: A ranking based approach. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 300–310.

[8] Yushun Dong, Oyku Deniz Kose, Yanning Shen, and Jundong Li. 2023. Fairness in Graph Machine Learning: Recent Advances and Future Prospectives. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 5794–5795.

[9] Yushun Dong, Ninghao Liu, Brian Jalaian, and Jundong Li. 2022. Edits: Modeling and mitigating data bias for graph neural networks. In *Proceedings of the ACM Web Conference 2022*. 1259–1269.

[10] Yushun Dong, Jing Ma, Song Wang, Chen Chen, and Jundong Li. 2023. Fairness in graph mining: A survey. *IEEE Transactions on Knowledge and Data Engineering* (2023).

[11] Yushun Dong, Binchi Zhang, Yiling Yuan, Na Zou, Qi Wang, and Jundong Li. 2023. Reliant: Fair knowledge distillation for graph neural networks. In *Proceedings of the 2023 SIAM International Conference on Data Mining (SDM)*. SIAM, 154–162.

[12] Matthias Fey and Jan E. Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

[13] Aric Hagberg, Pieter Swart, and Daniel S Chult. 2008. *Exploring network structure, dynamics, and function using NetworkX*. Technical Report. Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

[14] Charles R. Harris, K. Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array programming with NumPy. *Nature* 585 (2020), 357–362. https://doi.org/10.1038/s41586-020-2649-2

[15] Jian Kang, Jingrui He, Ross Maciejewski, and Hanghang Tong. 2020. Inform: Individual fairness on graph mining. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 379–389.

[16] Jian Kang, Yan Zhu, Yinglong Xia, Jiebo Luo, and Hanghang Tong. 2022. Rawlsgcn: Towards rawlsian difference principle on graph convolutional network. In *Proceedings of the ACM Web Conference 2022*. 1214–1225.

[17] Ahmad Khajehnejad, Moein Khajehnejad, Mahmoudreza Babaei, Krishna P Gummadi, Adrian Weller, and Baharan Mirzasoleiman. 2022. Crosswalk: Fairness-enhanced node representation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 11963–11970.

[18] Donald Loveland, Jiayi Pan, Aaresh Farrokh Bhathena, and Yiyang Lu. 2022. Fairedit: Preserving fairness in graph neural networks through greedy graph editing. *arXiv preprint arXiv:2201.03681* (2022).

[19] Jing Ma, Ruocheng Guo, Mengting Wan, Longqi Yang, Aidong Zhang, and Jundong Li. 2022. Learning fair node representations with graph counterfactual fairness. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 695–703.

[20] Wes McKinney et al. 2010. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, Vol. 445. Austin, TX, 51–56.

[21] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[22] Tahleen Rahman, Bartlomiej Surma, Michael Backes, and Yang Zhang. 2019. Fairwalk: Towards fair graph embedding. (2019).

[23] Tahleen Rahman, Bartlomiej Surma, Michael Backes, and Yang Zhang. 2019. Fairwalk: Towards Fair Graph Embedding. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 3289–3295. https://doi.org/10.24963/ijcai.2019/456

[24] Weihao Song, Yushun Dong, Ninghao Liu, and Jundong Li. 2022. Guide: Group equality informed individual fairness in graph neural networks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

[25] Guido Van Rossum, Barry Warsaw, and Nick Coghlan. 2001. PEP 8–style guide for python code. *Python. org* 1565 (2001), 28.

[26] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* 17 (2020), 261–272. https://doi.org/10.1038/s41592-019-0686-2

[27] Lijing Wang, Aniruddha Adiga, Jiangzhuo Chen, Adam Sadilek, Srinivasan Venkatramanan, and Madhav Marathe. 2022. Causalgnn: Causal-based graph neural networks for spatio-temporal epidemic forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 36. 12191–12199.

[28] Nan Wang, Lu Lin, Jundong Li, and Hongning Wang. 2022. Unbiased graph embedding with biased graph observations. In *Proceedings of the ACM Web Conference 2022*. 1423–1433.

[29] Yu Wang, Yuying Zhao, Yushun Dong, Huiyuan Chen, Jundong Li, and Tyler Derr. 2022. Improving fairness in graph neural networks via mitigating sensitive attribute leakage. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1938–1948.