EqGNN: Equalized Node Opportunity in Graphs

Uriel Singer, Kira Radinsky

Technion, Israel Institute of Technology
urielsinger@cs.technion.ac.il, kirar@cs.technion.ac.il

Abstract

Graph neural networks (GNNs), has been widely used for supervised learning tasks in graphs reaching state-of-the-art results. However, little work was dedicated to creating unbiased GNNs, i.e., where the classification is uncorrelated with sensitive attributes, such as race or gender. Some ignore the sensitive attributes or optimize for the criteria of statistical parity for fairness. However, it has been shown that neither approaches ensure fairness, but rather cripple the utility of the prediction task. In this work, we present a GNN framework that allows optimizing representations for the notion of *Equalized Odds* fairness criteria. The architecture is composed of three components: (1) a GNN classifier predicting the utility class, (2) a sampler learning the distribution of the sensitive attributes of the nodes given their labels. It generates samples fed into a (3) discriminator that discriminates between true and sampled sensitive attributes using a novel "permutation loss" function. Using these components, we train a model to neglect information regarding the sensitive attribute only with respect to its label. To the best of our knowledge, we are the first to optimize GNNs for the equalized odds criteria. We evaluate our classifier over several graph datasets and sensitive attributes and show our algorithm reaches state-of-the-art results.[1]

1 Introduction

Supervised learning was shown to exhibit bias depending on the data it was trained on (Pedreshi, Ruggieri, and Turini 2008). This problem is further amplified in graphs, where the graph topology was shown to exhibit different biases (Kang, Lijffijt, and Bie 2019; Singer, Radinsky, and Horvitz 2020). Many popular supervised-learning graph algorithms, such as graph neural networks (GNNs), employ message-passing with features aggregated from neighbors; which might further intensify this bias. For example, in social networks, communities are usually more connected between themselves. As GNNs aggregate information from neighbors, it makes it even harder for a classifier to realize the potential of an individual from a discriminated community.

Despite their success (Wu et al. 2020), little work has been dedicated to creating unbiased GNNs, where the classification is uncorrelated with sensitive attributes, such as race or gender. The little existing work, focused on ignoring the sensitive attributes (Obermeyer et al. 2019). However, "fairness through unawareness" has already been shown to predict sensitive attributes from other features (Barocas, Hardt, and Narayanan 2019). Others (Bose and Hamilton 2019; Dai and Wang 2021; Buyl and De Bie 2020; Rahman et al. 2019) focused on the criteria of Statistical Parity (SP) for fairness when training node embeddings, defined as follows:

Definition 1 (Statistical parity)**.** A predictor $\hat{\mathbf{Y}}$ satisfies *statistical parity* with respect to a sensitive attribute $\mathbf{A}$, if $\hat{\mathbf{Y}}$ and $\mathbf{A}$ are independent:

$$\hat{\mathbf{Y}} \perp\!\!\!\perp \mathbf{A} \tag{1}$$

Recently, Dwork et al. (2012) showed that SP does not ensure fairness and might actually cripple the utility of the prediction task. Consider the task of recommending a job to a person given their social network. Given a sensitive attribute of gender, SP will require the same amount of men and women per job. However, it allows accepting qualified applicants in one gender, but unqualified in another, as long as the percentages of acceptance match. This is not ideal because it does not consider the qualification of the person to a job. Lately, the notion of Equalized Odds (EO) was presented as an alternative fairness criteria (Hardt, Price, and Srebro 2016). Unlike SP, EO allows dependence on the sensitive attribute $\mathbf{A}$ but only through the target variable Y:

Definition 2 (Equalized odds)**.** A predictor $\hat{\mathbf{Y}}$ satisfies *equalized odds* with respect to a sensitive attribute $\mathbf{A}$, if $\hat{\mathbf{Y}}$ and $\mathbf{A}$ are independent conditional on the true label $\mathbf{Y}$:

$$\hat{\mathbf{Y}} \perp\!\!\!\perp \mathbf{A} \mid \mathbf{Y} \tag{2}$$

The definition encourages the use of features that allow to directly predict $\mathbf{Y}$, while not allowing to leverage $\mathbf{A}$ as a proxy for $\mathbf{Y}$. Consider our job acceptance example. For the outcome of $\mathbf{Y}$=Accept, we require $\hat{\mathbf{Y}}$ to have similar true and false positive rates across both genders. Notice that, $\hat{\mathbf{Y}} = \mathbf{Y}$ aligns with the EO constraint, but we enforce that the accuracy is equally high in both genders, and penalize models that have good performance on only one gender.

In this work, we present an architecture that optimizes graph classification for the EO criteria. Given a GNN classifier predicting a target class, we expand it with an EO regularization using a sampler and a discriminator components.

[1]GitHub repository with appendix, code, baselines, and data: https://github.com/urielsinger/EqGNN

The goal of the sampler component is to learn the distribution of the sensitive attributes of the nodes given their labels. The sampler generates examples that are then fed into a discriminator. The goal of the latter is to discriminate between true and sampled sensitive attributes. We present a novel loss function the discriminator minimizes – the *permutation loss*. Unlike cross-entropy loss, that compares two independent or unrelated groups, the permutation loss compares items under two separate scenarios – with sensitive attribute or with a generated balanced sensitive attribute.

We start by pretraining the sampler, and then train the discriminator along with the GNN classifier using adversarial training. This joint training allows the model to neglect information regarding the sensitive attribute only with respect to its label, as requested by the equalized odds fairness criteria. To the best of our knowledge, our work is the first to optimize GNNs for the equalized odds criteria.

The contributions of this work are fourfold: (1) We propose EqGNN, an algorithm with equalized odds regulation for graph classification tasks. (2) We propose a novel permutation loss which allows us to compare *pairs*. We use this loss in the special case of nodes in two different scenarios – one under the bias sensitive distribution, and the other under the generated unbiased distribution. (3) We empirically evaluate EqGNN on several real-world datasets and show superior performance to several baselines both in utility and in bias reduction. (4) We empirically evaluate the permutation loss over real-world datasets and show the importance of leveraging the pair information.

2 Related Work

Supervised learning in graphs has been applied in many applications, such as protein-protein interaction prediction (Grover and Leskovec 2016; Singer, Guy, and Radinsky 2019), human movement prediction (Yan, Xiong, and Lin 2018), traffic forecasting (Yu, Yin, and Zhu 2018; Cui et al. 2019) and other urban dynamics (Wang and Li 2017). Many supervised learning algorithms have been suggested for those tasks on graphs, including matrix factorization approaches (Belkin and Niyogi 2001; Tenenbaum, De Silva, and Langford 2000; Yan et al. 2006; Roweis and Saul 2000), random walks approaches (Perozzi, Al-Rfou, and Skiena 2014; Grover and Leskovec 2016) and graph neural network, which recently showed state-of-the-art results on many tasks (Wu et al. 2020). The latter is an adaptation of neural networks to the graph domain. GNNs create different differential layers that can be added to many different architectures and tasks. GNNs utilize the graph structure by propagating information through the edges and nodes. For instance, GCN (Kipf and Welling 2017) and graphSAGE (Hamilton, Ying, and Leskovec 2017) update the nodes representation by averaging over the representations of all neighbors, while Veličković et al. (2017) proposed an attention mechanism to learn the importance of each specific neighbor.

Fairness in graphs was mostly studied in the context of group fairness, by optimizing the SP fairness criteria. Rahman et al. (2019) creates fair random walks by first sampling a sensitive attribute and only then sampling a neighbor from those who hold that specific sensitive attribute. For instance, if most nodes represent men while the minority represent women, the fair random walk promises that the presence of men and women in the random walks will be equal. Buyl and De Bie (2020) proposed a Bayesian method for learning embeddings by using a biased prior. Others, focus on unbiasing the graph prediction task itself rather than the node embeddings. For example, Bose and Hamilton (2019) use a set of adversarial filters to remove information about predefined sensitive attributes. It is learned in a self supervised way by using a graph-auto-encoder to reconstruct the graph edges. Dai and Wang (2021) offer a discriminator that discriminates between the nodes sensitive attributes. In their setup, not all nodes sensitive attributes are known, and therefore, they add an additional component that predicts the missing attributes. Kang et al. (2020) tackle the challenge of individual fairness in graphs. In this work, we propose a GNN framework optimizing the EO fairness criteria. To the best of our knowledge, our work is the first to study fairness in graphs in the context of EO fairness.

3 Equalized-Odds Fair GNN

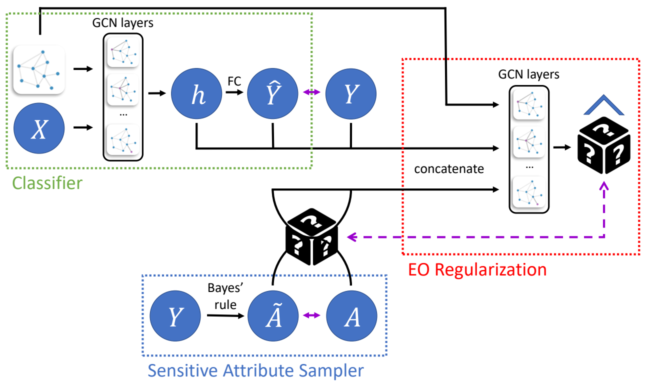

Figure 1: The full EqGNN architecture. The blue box represents the sampler model that given a label, samples a *dummy sensitive attribute* (Section 3.1 for details). This model is pretrained independently. The green box represents the classifier, which given a graph and node features, tries to predict its label. The red box represents the discriminator, which minimizes a permutation loss (Section 3.2). Purple arrows represent loss functions while the magic box represents a random bit (0 or 1), which is used for shuffling the sensitive attribute with its dummy.

Let $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ be a graph, where $\mathbf{E}$ is the list of edges, $\mathbf{V}$ the list of nodes, $\mathbf{Y}$ the labels, and $\mathbf{A}$ the sensitive attributes. Each node is represented via n features. We denote $X^{|V| \times n}$ as the feature matrix for the nodes in $\mathbf{V}$. Our goal is to learn a function $F(\cdot)$ with parameters θ_F, that given a graph $\mathbf{G}$, maps a node $v \in \mathbf{V}$ represented by a feature vector, to its label. In this work, we present an architecture that can leverage any graph neural network classifier. For simplicity, we consider a simple GNN architecture for $F(\cdot)$ as suggested by Kipf and Welling (2017): we define $F(\cdot)$ to be two GCN layers, outputting a hidden representation $\mathbf{h}$ for each node. This representation then enters a fully connected layer that outputs $\hat{\mathbf{Y}}$. The GNN optimization goal is to minimize the distance between $\hat{\mathbf{Y}}$ and $\mathbf{Y}$ using a loss function ℓ. ℓ can be categorical cross-entropy (CCE) for multi-class classification, binary cross-entropy (BCE) for binary classification, or mean square error (L2) for regression problems. We extend the optimization to $\min_{\theta_F} L_{task} = \ell(\hat{\mathbf{Y}}, \mathbf{Y})$ while satisfying Eq. 2 for fair prediction.

We propose a method, EqGNN, that trains a GNN model to neglect information regarding the sensitive attribute only with respect to its label. The full architecture of EqGNN is depicted in Figure 1. Our method pretrains a sampler (Section 3.1), to learn the distribution $\mathbb{P}_{\mathbf{A}|\mathbf{Y}}$ of the sensitive attributes of the nodes given their labels (marked in blue in Figure 1). We train a GNN classifier F (marked in green in Figure 1), while regularizing it with a discriminator that discriminates between true and sampled sensitive attributes (marked in red in Figure 1) . Section 3.2 presents the EO regularization. The regularization is done using a novel loss function – the "permutation loss", which is capable of comparing paired samples. For the unique setup of adversarial learning over graphs, we show that incorporating the permutation loss in a discriminator, brings performance gains both in utility and in EO. Section 3.3 presents the full EqGNN model optimization procedure.

3.1 Sensitive Attribute Sampler

To comply with the SP criteria (Eq. 1), given a sample i, we wish the prediction of the classifier, $\hat{\mathbf{Y}}_i$, to be independent of the sample's sensitive attribute $\mathbf{A}_i$. In order to check if this criteria is kept, we can sample a fake attribute out of $\mathbb{P}_{\mathbf{A}}$ (e.g., in case of equal sized groups, a random attribute), and check if $\hat{\mathbf{Y}}_i$ can predict the true or fake attribute. If it is not able to predict, this means that $\hat{\mathbf{Y}}_i$ is independent of $\mathbf{A}_i$ and the SP criteria is kept. As all information of $\hat{\mathbf{Y}}_i$ is also represented in the hidden representation $\mathbf{h}_i$, one can simply train a discriminator to predict the sensitive attribute given the hidden representation $\mathbf{h}_i$. A similar idea was suggested by Dai and Wang (2021); Bose and Hamilton (2019).

To comply with the EO criteria (Eq. 2), the classifier should not be able to separate between an example with a real sensitive attribute $\mathbf{A}_i$ and an example with an attribute sampled from the conditional distribution $\mathbb{P}_{\mathbf{A}|\mathbf{Y}}(\mathbf{A}_i \mid \mathbf{Y}_i)$. Therefore, we jointly train the classifier with a discriminator that learns to separate between the two examples. Formally, given a sample i, we would want the prediction of the classifier, $\hat{\mathbf{Y}}_i$, to be independent of the attribute $\mathbf{A}_i$ only given its label $\mathbf{Y}_i$. Thus, instead of sampling the fake attribute out of $\mathbb{P}_{\mathbf{A}}$ we sample the fake attribute out of $\mathbb{P}_{\mathbf{A}|\mathbf{Y}}$.

We continue describing the sensitive attribute distribution learning process, $\mathbb{P}_{\mathbf{A}|\mathbf{Y}}$, and then present how the model samples *dummy sensitive attributes* that will be used as "negative" examples for the discriminator.

Sensitive Attribute Distribution Learning Our goal is to learn the distribution $\mathbb{P}_{\mathbf{A}|\mathbf{Y}}$. For a specific sensitive attribute a and label y, the probability can be expressed using Bayes' rule as:

$$\mathbb{P}_{\mathbf{A}|\mathbf{Y}}(\mathbf{A} = a \mid \mathbf{Y} = y) = \frac{\mathbb{P}(\mathbf{Y} = y \mid \mathbf{A} = a)\mathbb{P}(\mathbf{A} = a)}{\sum_{a' \in \mathbf{A}} \mathbb{P}(\mathbf{Y} = y \mid \mathbf{A} = a')\mathbb{P}(\mathbf{A} = a')} \tag{3}$$

The term $\mathbb{P}(\mathbf{Y} = y \mid \mathbf{A} = a)$ can be derived from the data by counting the number of samples that are both with label y and sensitive attribute a, divided by the number of samples with sensitive attribute a. Similarly, $\mathbb{P}(\mathbf{A} = a)$ is calculated as the number of samples with sensitive attribute a, divided by the total number of samples. In a regression setup, these can be approximated using a linear kernel density estimation.

Fair Dummy Attributes During training of the end-to-end model (Section 3.3), the sampler receives a training example $(\mathbf{X}_i, \mathbf{Y}_i, \mathbf{A}_i)$ and generates a *dummy attribute* by sampling $\tilde{\mathbf{A}}_i \sim \mathbb{P}_{\mathbf{A}|\mathbf{Y}}(\mathbf{A}_i \mid \mathbf{Y}_i)$. Notice that $\mathbf{A}_i$ and $\tilde{\mathbf{A}}_i$ are equally distributed given $\mathbf{Y}_i$. This ensures that if the classifier satisfies the EO criteria, then $(\mathbf{X}_i, \mathbf{Y}_i, \mathbf{A}_i)$ and $(\mathbf{X}_i, \mathbf{Y}_i, \tilde{\mathbf{A}}_i)$ will receive an identical classification, whereas otherwise it will result in different classifications. See Section 3.2 for further explanation of the optimization process that utilizes the *dummy attributes* for regularizing the classifier for EO.

3.2 EO Regularization

A GNN classifier without regularization might learn to predict biased node labels based on their sensitive attributes. To satisfy EO, the classifier should be unable to distinguish between real examples and generated examples with *dummy sensitive attributes*. Therefore, we utilize an adversarial learning process and add a discriminator that learns to distinguish between real and fake examples with *dummy sensitive attributes*.

The discriminator receives two types of examples: (1) real examples $(\mathbf{Y}_i, \mathbf{A}_i, \hat{\mathbf{Y}}_i)$ and (2) negative examples $(\mathbf{Y}_i, \tilde{\mathbf{A}}_i, \hat{\mathbf{Y}}_i)$, where $\tilde{\mathbf{A}}_i \sim \mathbb{P}_{\mathbf{A}|\mathbf{Y}}(\mathbf{A}_i \mid \mathbf{Y}_i)$ is generated by the pretrained sampler. The discriminator learns a function $D(\cdot)$ with parameters θ_D, that given a sample, classifies it to be true or fake. The classifier in its turn tries to "fool" it. This ensures the classifier doesn't hold bias towards specific labels and sensitive attributes and keeps the EO criterion. Intuitively, as the classifier tries to fool the discriminator, one might consider $\mathbf{h}_i$, the last hidden layer of the classifier $F(\cdot)$, as the unbiased representation of node i. Therefore, we concatenate the unbiased representation of the node, $\mathbf{h}_i$, to the samples the discriminator receives.

As a result, the formal adversarial loss is defined as:

$$\begin{aligned}\min_{\theta_F} \max_{\theta_D} L_{adv} &= \mathbb{E}\left[log(D(\mathbf{Y} \mid\mid \mathbf{A} \mid\mid F(\mathbf{X}, \mathbf{G}) \mid\mid \mathbf{h}))\right] \\ &+ \mathbb{E}_{\tilde{\mathbf{A}} \sim \mathbb{P}_{\mathbf{A}|\mathbf{Y}}(\mathbf{A}|\mathbf{Y})}\left[log(1 - D(\mathbf{Y} \mid\mid \tilde{\mathbf{A}} \mid\mid F(\mathbf{X}, \mathbf{G}) \mid\mid \mathbf{h}))\right]\end{aligned} \tag{4}$$

where $\mathbb{E}$ is the expected value, and $||$ represents the concatenation operator. The discriminator tries to maximize L_{adv} to distinguish between true and fake samples, while the classifier tries to minimize it, in order to "fool" him. In our implementation, $D(\cdot)$ is a GNN with two GCN layers outputting the probability of being a true sample.

We observe that true and fake attributes are paired (the same node, with a real sensitive attribute or a *dummy attribute*). A binary loss as defined in Eq. 4 holds for unpaired examples, and does not take advantage of knowing the fake and true attributes are paired. Therefore, we upgrade the loss to handle paired nodes by utilizing the permutation loss. We first formally define and explain the permutation loss in the following section, and then continue discussing its implementation details in our architecture.

Permutation Loss In this section, we formally define the new permutation loss test presented in this work. Let us assume $\mathbf{X}_1$ and $\mathbf{X}_2$ are two groups of subjects. Many applications are interested of learning and understanding the difference between the two. For example, in the case where $\mathbf{X}_1$ represents test results of students in one class, while $\mathbf{X}_2$ represents test results of a different class, it will be interesting to check if the two classes are equally distributed (i.e., $\mathbb{P}(\mathbf{X}_1) \sim \mathbb{P}(\mathbf{X}_2)$).

Definition 3 (T-test)**.** Given two groups $\mathbf{X}_1, \mathbf{X}_2 \subset \mathbb{R}$, the statistical difference can be measured by the t-statistic:

$$t = \frac{\bar{\mathbf{X}}_1 - \bar{\mathbf{X}}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

where $\bar{\mathbf{X}}_i$, s_i, and n_i are the means, variances, and group sizes respectively.

While this test assumes $\mathbf{X}_1, \mathbf{X}_2$ are scalars that are normally distributed, Lopez-Paz and Oquab (2017) proposed a method called C2ST that handles cases where $\mathbf{X}_1, \mathbf{X}_2 \subset \mathbb{R}^d$ for $d \geq 1$. They proposed a classifier that is trained to predict the correct group of a given sample, which belongs to either X_1 or X_2. By doing so, given a test-set, they are able to calculate the t-statistic by simply checking the number of correct predictions:

Definition 4 (C2ST)**.** Given two groups $\mathbf{X}_1, \mathbf{X}_2 \subset \mathbb{R}^d$ (labeled 0 and 1 respectively), the statistical difference can be measured by the t-statistic (Lopez-Paz and Oquab 2017):

$$t = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\left[\mathbb{I}\left(f(x_i) > \frac{1}{2}\right) = y_i\right]$$

where $x_i \in \mathbf{X}_1 \cup \mathbf{X}_2$, y_i is x_i's original group label, $N = |\mathbf{X}_1| + |\mathbf{X}_2|$ is the number of samples in the test-set, and f is a trained classifier outputting the probability a sample is sampled from group 1.

The basic (and yet important) idea behind this test is that if a classifier is not able to predict the group of a given sample, then the groups are equality distributed. Mathematically, the t-statistic can be calculated by the number of correct samples of the classifier.

However, the C2ST criterion is not optimal when the samples are paired ($\mathbf{X}_1^i$ and $\mathbf{X}_2^i$ represent the same subject), as it doesn't leverage the paired information. Consider the following example: assuming the pairs follow the following rule: $\mathbf{X}_1 \sim \mathcal{N}_d(\mathbf{0}, \mathbf{1})$ and $\mathbf{X}_2^i = \mathbf{X}_1^i + \epsilon \quad \forall \mathbf{X}_2^i \in \mathbf{X}_2$. As the two Gaussians overlap, a simple linear classifier will not be able to detect any significant difference between the two groups, while we hold the information that there is a significant difference ($\mathbf{X}_1^i$ is always smaller in ϵ from its pair $\mathbf{X}_2^i$). Therefore, it is necessary to define a new test that can manage paired examples.

Definition 5 (Paired T-test)**.** Given two paired groups $\mathbf{X}_1, \mathbf{X}_2 \subset \mathbb{R}$, and $\mathbf{X}_D = \mathbf{X}_1 - \mathbf{X}_2$, the statistical difference can be measured by the t-statistic:

$$t = \bar{\mathbf{X}}_D / \frac{s_D}{\sqrt{n}}$$

where $\bar{\mathbf{X}}_D$, s_D are the mean and standard deviation of $\mathbf{X}_D$, and $n = |\mathbf{X}_D|$ is the number of pairs.

Again, this Paired T-test assumes $\mathbf{X}_1, \mathbf{X}_2$ are scalars that are normally distributed. A naive adaptation of Lopez-Paz and Oquab (2017) for paired data with $d \geq 1$, would be to first map the pairs into scalars and then calculate their differences (as known in the paired student t-test for $d = 1$). This approach also assumes the samples are normally distributed, and therefore is not robust enough. An alternative to the paired t-test is the permutation test (Odén, Wedel et al. 1975), which has no assumptions on the distribution. It checks how different the t-statistic of a specific permutation is from many (or all) other random permutations t-statistics. By doing so, it is able to calculate the p-value of that specific permutation. We suggest a differential version of the permutation test. This is done by using a neural network architecture that receives either a real permutation or a shuffled, and tries to predict the true permutation. We draw the reader attention that other unpaired distribution distances, e.g. KL-divergence or Wasserstein, are not applicable for our problem. Those distances are calculated over distributions and not over pair of values.

The permutation phase consists of four steps: (1) For each pair, $\mathbf{X}_1^i, \mathbf{X}_2^i$ each of size d, sample a random number $l_i \in \{0, 1\}$. (2) If $l_i = 0$, concatenate the pair in the original order: $[\mathbf{X}_1^i, \mathbf{X}_2^i]$, while if $l_i = 1$, concatenate in the permuted order: $[\mathbf{X}_2^i, \mathbf{X}_1^i]$. This resolves us with a vector of size $2d$. (3) This sample enters a classifier that tries to predict l_i using binary-cross-entropy . (4) We update the classifier weights, and return to step 1 until convergence. Assuming $\mathbf{X}_1$ and $\mathbf{X}_2$ are exchangeable, the classifier will not be able to distinguish if a permutation was performed or not. The idea behind this test is that if a classifier is not able to predict the true ordering of the pairs, it means that there is no significant difference between this specific permutation or any other permutation. Mathematically, similarly to C2ST, the t-statistic can be calculated by the number of correct samples of the classifier. See Appendix A for the full differential permutation loss algorithm. While this is similar to the motivation of the naive permutation test (Odén, Wedel et al. 1975), we offer additional benefits: (1) The test is differential, meaning we can represent it using a neural network as a layer (see Section 3.2). (2) The test can handle $d \geq 1$. (3) Given a use-case, we do not need to define a specific t-statistic as required by the naive permutation test, but rather the classifier checks for any possible signal.

As a real life example that explains the power of the loss, assume a person is given a real coin and fake coin. While she observes each one separately, her confidence of which is which, will be much less if she would rather receive them together. This real life example demonstrates the important difference between C2ST and the permutation loss (see Appendix B for experiments for this example and other experiments over synthetic data).

Permutation Loss for EO Regularization Going back to our discriminator, we observe that true and fake attributes are paired (the same node, with a real sensitive attribute or a *dummy sensitive attribute*). We create paired samples: $(\mathbf{Y}_i, \mathbf{A}_i, \tilde{\mathbf{A}}_i, \hat{\mathbf{Y}}_i, \mathbf{h}_i)$, and at each step we randomly permute the sensitive attribute and its dummy attribute, creating a sample labeled as permuted or not. The result of this process are samples $(\mathbf{Y}_i, \mathbf{A}_i, \tilde{\mathbf{A}}_i, \hat{\mathbf{Y}}_i, \mathbf{h}_i)$ with label 0 indicating no permutation was applied, or $(\mathbf{Y}_i, \tilde{\mathbf{A}}_i, \mathbf{A}_i, \hat{\mathbf{Y}}_i, \mathbf{h}_i)$ with label 1 indicating a permutation was applied. The discriminator therefore receives the samples and predicts the probability of whether a permutation was applied. We therefore adapt the adversarial loss of Eq. 4 to:

$$\begin{aligned}&\min_{\theta_F}\max_{\theta_D} L_{adv} = \\ &\mathbb{E}_{\tilde{\mathbf{A}}\sim\mathbb{P}_{\mathbf{A}|\mathbf{Y}}(\mathbf{A}|\mathbf{Y})}\Big[log(D(\mathbf{Y} \,||\, \tilde{\mathbf{A}} \,||\, \mathbf{A} \,||\, F(\mathbf{X},\mathbf{G}) \,||\, \mathbf{h}))\Big] + \\ &\mathbb{E}_{\tilde{\mathbf{A}}\sim\mathbb{P}_{\mathbf{A}|\mathbf{Y}}(\mathbf{A}|\mathbf{Y})}\Big[log(1 - D(\mathbf{Y} \,||\, \mathbf{A} \,||\, \tilde{\mathbf{A}} \,||\, F(\mathbf{X},\mathbf{G}) \,||\, \mathbf{h}))\Big]\end{aligned} \tag{5}$$

The loss used in the permutation test is binary cross-entropy and therefore convex. As an additional regularization and to improve stability of the classifier, similarly to Romano, Bates, and Candès (2020), we propose to minimize the absolute difference between the covariance of $\hat{\mathbf{Y}}$ and $\mathbf{A}$ from the covariance of $\hat{\mathbf{Y}}$ and $\tilde{\mathbf{A}}$:

$$\min_{\theta_F} L_{cov} = \|cov(\hat{\mathbf{Y}}, \mathbf{A}) - cov(\hat{\mathbf{Y}}, \tilde{\mathbf{A}})\|^2 \tag{6}$$

3.3 Full EqGNN Architecture

The sampler is pretrained using Eq. 3. We then jointly train the classifier and discriminator by optimizing the objective:

$$\min_{\theta_F}\max_{\theta_D} L_{task} + \lambda(L_{adv} + \gamma L_{cov}) \tag{7}$$

where θ_F are the parameters of the classifier and θ_D are the parameters of the discriminator. λ and γ are hyperparameters that are used to tune the different regulations. This objective is then optimized for θ_F and θ_D one step at a time using the Adam optimizer (Kingma and Ba 2015), with learning rate 10^{-3} and weight-decay 10^{-5}. The training procedure is further detailed in Appendix C.

4 Experimental Setup

In this section, we describe our datasets, baselines, and metrics. Our baselines include fair baselines designed specifically for graphs, and general fair baselines that we adapted to the graph domain.

4.1 Datasets

Table 1 summarizes the datasets' characteristics used for our experiments. Intra-group edges are the edges between similar sensitive attributes, while inter-group edges are edges between different sensitive attributes.

Dataset	Pokec-region	Pokec-gender	NBA
# of nodes	$67,796$	$67,796$	355
# of attributes	276	276	95
# of edges	$617,958$	$617,958$	$9,477$
sensitive groups ratio	1.84	1.02	3.08
# of inter-group edges	$30,519$	$339,461$	$2,472$
# of intra-group edges	$587,439$	$278,497$	$7,005$

Table 1: Datasets' characteristics.

Pokec (Takac and Zabovsky 2012). Pokec is a popular social network in Slovakia. An anonymized snapshot of the network was taken in 2012. User profiles include gender, age, hobbies, interest, education, etc. The original Pokec dataset contains millions of users. We sampled a subnetwork of the "Zilinsky" province. We create two datasets, where the sensitive attribute in one is the gender, and region in the other. The label used for classification is the job of the user. The job field was grouped in the following way: (1)"education" and "student", (2)"services & trade" and "construction", and (3) "unemployed".

NBA (Dai and Wang 2021) This dataset was presented in the FairGNN baseline paper. The NBA Kaggle dataset contains around 400 basketball players with features including performance statistics, nationality, age, etc. This dataset was extended in (Dai and Wang 2021) to include the relationships of the NBA basketball players on Twitter. The binary sensitive attribute is whether a player is a U.S. player or an overseas player, while the task is to predict whether a salary of the player is over the median.

4.2 Baselines

In our evaluation, we compare to a classic baseline (GCN), an EO optimized baseline (Debias), and a SP optimized baseline (FairGNN):

No-Fairness Optimization Baseline: **GCN** (Kipf and Welling 2017) is a classic GNN layer that updates a node representation by averaging the representations of its neighbors. For fair comparison, we implemented GCN as the classifier of the EqGNN architecture (i.e., an unregulated EqGNN model, with the only difference of $\lambda = 0.$).

EO Optimization Baseline: **Debias** (Zhang, Lemoine, and Mitchell 2018): optimizes EO by using a discriminator that given $\mathbf{Y}$ and $\hat{\mathbf{Y}}$ predicts the sensitive attribute. While Debias is a non-graph architecture, for fair comparison, we implemented Debias with the exact architecture as EqGNN. Unlike EqGNN, Debias's discriminator receives as input only Y and $\hat{Y}$ (without the sensitive attribute or *dummy attribute*) and predicts the sensitive attribute. As the discriminator receives $\mathbf{Y}$, it neglects the sensitive information with respect to $\mathbf{Y}$ and, therefore optimizes for EO.

SP Optimization Baseline: **FairGNN** (Dai and Wang 2021) optimizes SP by using a discriminator that, given $\mathbf{h}$, predicts the sensitive attribute. By doing so, they neglect the sensitive information from $\mathbf{h}$. As this is without respect to $\mathbf{Y}$, they optimize SP (further explained in Section 3.1). FairGNN offers an additional predictor for nodes with unknown sensitive attributes. As our setup includes all nodes' sensitive attributes, this predictor is irrelevant. We opted to use FairGCN for fair comparison. In addition, we generalized their architecture to support multi-class classification.

For all baselines, 50% of nodes are used for training, 25% for validation and 25% for testing. The validation set is used for choosing the best model for each baseline throughout the training. As the classifier is the only part of the architecture used for testing, an early stopping was implemented after its validation loss (Eq. 7) hasn't improved for 50 epochs. The epoch with the best validation loss was then used for testing. All results are averaged over 20 different train/validation/test splits for Pokec datasets and 40 for the NBA dataset. For fair comparison, we implemented grid-search for all baselines over $\lambda \in \{0.01, 0.1, 1, 10\}$ for baselines with a discriminator, and $\gamma \in \{0, 50\}$ for baselines with a covariance expression. For both Pokec datasets and for all baselines $\lambda = 1$ and $\gamma = 50$, while for NBA we end up using $\lambda = 0.1$ and $\gamma = 50$ expect for FairGNN with $\lambda = 0.01$. All experiments used a single Nvidia P100 GPU with the average run of 5 minutes per seed for Pokec and 1 minute for NBA.

As the training procedure has not changed drastically compared to the different baselines (please refer to Section 3.3), the time and memory complexity as compared to the baselines remains similar.

4.3 Metrics

Fairness Metrics We strive to create a metric for SP (EO). The definition in Eq. 1 (Eq. 2), can be formally written as:

$$\begin{aligned} &\mathbb{P}(\hat{\mathbf{Y}} = y | \mathbf{A} = a_1) = \mathbb{P}(\hat{\mathbf{Y}} = y | \mathbf{A} = a_2) \\ &(\mathbb{P}(\hat{\mathbf{Y}} = y | \mathbf{Y} = y, \mathbf{A} = a_1) = \mathbb{P}(\hat{\mathbf{Y}} = y | \mathbf{Y} = y, \mathbf{A} = a_2)) \end{aligned} \tag{8}$$

Where $a_1, a_2 \in \mathbf{A}$ and $y \in \mathbf{Y}$. The values of $\mathbb{P}(\hat{\mathbf{Y}} = y | \mathbf{A} = a)$ $(\mathbb{P}(\hat{\mathbf{Y}} = y | \mathbf{Y} = y, \mathbf{A} = a))$ are calculated from the test-set as follows: given all samples with sensitive attribute a (and label y), we calculate the proportion of samples that where labeled y by the model. As we handle a binary sensitive attribute, given a label $y \in \mathbf{Y}$, we calculate the absolute difference between the two sensitive attribute values:

$$\begin{aligned} &\Delta SP(y) = |\mathbb{P}(\hat{\mathbf{Y}} = y | \mathbf{A} = 0) - \mathbb{P}(\hat{\mathbf{Y}} = y | \mathbf{A} = 1)| \\ &(\Delta EO(y) = \\ &|\mathbb{P}(\hat{\mathbf{Y}} = y | \mathbf{Y} = y, \mathbf{A} = 0) - \mathbb{P}(\hat{\mathbf{Y}} = y | \mathbf{Y} = y, \mathbf{A} = 1)|) \end{aligned} \tag{9}$$

According to Eq. 8, our goal is to have both probabilities equal. Therefore, we desire $\Delta SP(y)$ $(\Delta EO(y))$ to strive to 0. As in most cases, most classes can be fairly classified, we are interested in measuring the worst class fairness performance, therefore, we aggregate $\Delta SP(y)$ $(\Delta EO(y))$ for all labels using the max operator to get a final scalar metric:

$$\begin{aligned} &\Delta SP = max(\{\Delta SP(y) | y \in \mathbf{Y}\}) \\ &(\Delta EO = max(\{\Delta EO(y) | y \in \mathbf{Y}\})) \end{aligned} \tag{10}$$

As we propose an equalized odds architecture, ΔEO is our main fairness metric.

Performance Metrics As our main classification metric, we used the F1 score. We examined both the micro F1 score, which is computed globally based on the true and false predictions, and the macro F1 score, computed per each class and averaged across all classes. For completeness, we also report the Accuracy (ACC).

Dataset	Metrics	No-Fairness	SP Optimization	EO Optimization	
		GCN	FairGNN	Debias	EqGNN
Pokec-gender	ΔEO (%)	12.3 ± 1.0	13.7 ± 1.1	11.1 ± 0.7	$\mathbf{9.4 \pm 0.8}$
	ΔSP (%)	8.4 ± 0.6	10.2 ± 0.4	8.1 ± 0.6	$\mathbf{4.2 \pm 0.3}$
	ACC (%)	65.1 ± 0.2	65.2 ± 0.2	65.0 ± 0.2	65.4 ± 0.2
	F1-macro (%)	61.2 ± 0.2	61.3 ± 0.2	61.1 ± 0.2	61.3 ± 0.2
	F1-micro (%)	66.5 ± 0.2	66.6 ± 0.2	66.5 ± 0.2	66.9 ± 0.2
Pokec-region	ΔEO (%)	17.3 ± 1.2	17.2 ± 1.2	15.8 ± 1.1	$\mathbf{11.2 \pm 1.3}$
	ΔSP (%)	8.4 ± 0.4	8.2 ± 0.5	7.1 ± 0.4	$\mathbf{4.1 \pm 0.5}$
	ACC (%)	64.9 ± 0.2	64.1 ± 0.3	63.1 ± 0.3	65.1 ± 0.3
	F1-macro (%)	61.0 ± 0.2	60.4 ± 0.2	59.6 ± 0.2	61.3 ± 0.2
	F1-micro (%)	66.4 ± 0.2	65.4 ± 0.3	64.4 ± 0.3	66.5 ± 0.3
NBA	ΔEO (%)	25.9 ± 2.0	24.9 ± 1.9	24.0 ± 1.6	$\mathbf{22.1 \pm 2.1}$
	ΔSP (%)	9.3 ± 1.4	8.0 ± 1.1	8.0 ± 1.0	$\mathbf{7.0 \pm 1.2}$
	ACC (%)	72.9 ± 0.8	71.6 ± 1.0	72.9 ± 0.7	72.7 ± 0.9
	F1-macro (%)	72.7 ± 0.8	70.6 ± 1.3	72.3 ± 0.7	72.2 ± 0.9
	F1-micro (%)	72.8 ± 0.8	70.8 ± 1.4	72.6 ± 0.7	72.4 ± 0.9

Table 2: Fairness and performance results

5 Experimental Results

In this section, we compare EqGNN to the baselines over numerous datasets (Section 5.1). We then demonstrate the importance of λ (Section 5.2) and the permutation loss (Section 5.3) to the EqGNN architecture. For comparison over synthetic datasets and qualitative examples see Appendix D.

5.1 Main Result

Table 2 reports the results of EqGNN and baselines over the datasets with respect to the performance and fairness metrics. We can notice that, while the performance metrics are very much similar between all baselines (apart from Debias in Pokec-region), EqGNN outperforms all other baselines in both fairness metrics. An interesting observation is that Debias is the second best, after EqGNN, to improve the EO metric, without harming the performance metrics. This can be explained as it is the only baseline to optimize with respect to EO. Additionally, Debias has gained fairness in Pokec-region, but at the cost of performance. This is a general phenomena: the lower the performance, the better the fairness. For example, when the performance is random, surely the algorithm doesn't prefer any particular group and therefore is extremely fair. Here, EqGNN is able to both optimize the fairness metrics while keeping the performance metrics high. The particularly low performance demonstrated by FairGNN was also reproduced on the authors original code and datasets using different seeds. Surprisingly, EqGNN outperforms both SP and EO. This can be explained as while in the binary case there is a trade-off between the two metrics, there is no theoretical trade-off in the case of multi-class (Barocas, Hardt, and Narayanan 2019).

5.2 The Discriminator for Bias Reduction

As a second analysis, we demonstrate the importance of the λ parameter with respect to the performance and fairness metrics. The λ hyper-parameter serves as a regularization for task performance as opposed to fairness. High values of λ cause the discriminator EO regulation on the classifier to be higher. While EqGNN results reported in this paper use $\lambda = 1$ for Pokec datasets, and $\lambda = 0.1$ for NBA, we show additional results for $\lambda \in \{0.01, 0.1, 1, 10, 100\}$. In Figure 2, we can observe that the selected λs show best results over all metrics for Pokec-gender, while similar results

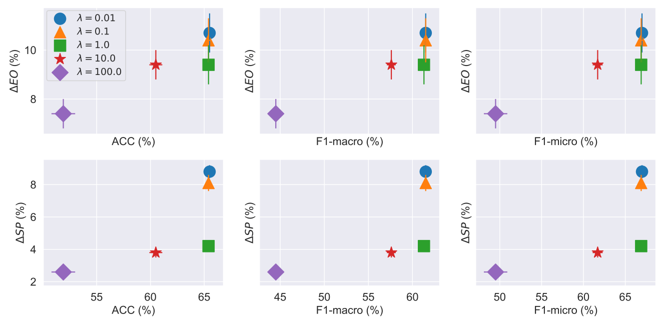

Figure 2: The comparisons of different λ values over the Pokec-gender dataset. Lower-right is better.

Dataset	Metrics	Unpaired	Paired	Permutation/h	Permutation
Pokec-gender	ΔEO (%)	10.9 ± 1.1	10.5 ± 0.6	11.4 ± 0.9	$\mathbf{9.4 \pm 0.8}$
	ΔSP (%)	5.8 ± 0.8	3.9 ± 0.4	7.2 ± 1.1	4.2 ± 0.3
	ACC (%)	65.2 ± 0.4	62.5 ± 0.7	66.0 ± 0.4	65.4 ± 0.2
	F1-macro (%)	61.1 ± 0.3	58.6 ± 0.6	61.7 ± 0.3	61.3 ± 0.2
	F1-micro (%)	66.7 ± 0.4	63.8 ± 0.7	67.5 ± 0.3	66.9 ± 0.2
Pokec-region	ΔEO (%)	15.5 ± 2.1	13.5 ± 2.1	16.5 ± 1.8	$\mathbf{11.2 \pm 1.3}$
	ΔSP (%)	6.4 ± 0.6	4.2 ± 0.7	7.9 ± 0.5	4.1 ± 0.5
	ACC (%)	64.2 ± 0.4	62.9 ± 0.5	65.2 ± 0.2	65.1 ± 0.3
	F1-macro (%)	60.3 ± 0.3	59.3 ± 0.4	61.2 ± 0.2	61.3 ± 0.2
	F1-micro (%)	65.6 ± 0.4	64.2 ± 0.5	66.7 ± 0.2	66.5 ± 0.3
NBA	ΔEO (%)	22.7 ± 1.9	25.1 ± 1.8	22.4 ± 1.9	$\mathbf{22.1 \pm 2.1}$
	ΔSP (%)	7.5 ± 0.8	7.3 ± 1.1	7.1 ± 0.9	7.0 ± 1.2
	ACC (%)	70.9 ± 1.1	72.7 ± 0.7	72.0 ± 0.9	72.7 ± 0.9
	F1-macro (%)	68.8 ± 1.7	71.9 ± 0.8	70.9 ± 1.2	72.2 ± 0.9
	F1-micro (%)	69.2 ± 1.7	72.2 ± 0.8	71.2 ± 1.2	72.4 ± 0.9

Table 3: The comparisons of different loss functions.

where shown over Pokec-region and NBA. Obviously, enlarging the λ results in a more fair model but at the cost of performance. The λ hyper-parameter is an issue of priority: depending on the task, one should decide what should be the performance vs. fairness prioritization. Therefore, EqGNN can be used with any desired λ where we chose $\lambda = 1$ as it is the elbow of the curve.

5.3 The Importance of the Permutation Loss

As an ablation study, we compare different loss functions for the discriminator. We choose to compare the permutation loss with three different loss functions: (1) Unpaired: Inspired by Romano, Bates, and Candès (2020), an unpaired binary cross-entropy loss as presented in Eq. 4. The loss is estimated by a classifier that predicts if a sample represents a real sensitive attribute or a dummy. (2) Permutation/h: A permutation loss without concatenating the hidden representation $\mathbf{h}_i$ to the discriminator samples, while leaving the sample to be $(\mathbf{Y}_i, \mathbf{A}_i, \tilde{\mathbf{A}}_i, \hat{\mathbf{Y}}_i)$. (3) Paired: A paired loss:

$$\begin{aligned} \min_{\theta_F} \max_{\theta_D} L_{adv} = \\ \mathbf{E}_{\tilde{\mathbf{A}} \sim \mathbb{P}_{\mathbf{A}|\mathbf{Y}}(\mathbf{A}|\mathbf{Y})} [\sigma(D(\mathbf{Y} \mid\mid \mathbf{A} \mid\mid F(\mathbf{X}, \mathbf{G}) \mid\mid \mathbf{h})) \\ - \sigma(D(\mathbf{Y} \mid\mid \tilde{\mathbf{A}} \mid\mid F(\mathbf{X}, \mathbf{G}) \mid\mid \mathbf{h}))] \end{aligned} \tag{11}$$

where σ is the Sigmoid activation. This loss is the known paired student t-test with a neural network version of it (as demonstrated by Lopez-Paz and Oquab (2017) on the unpaired t-test). Implementation of this loss when summing the absolute differences yielded poor results. We therefore, report a version of this loss with a summation over non absolute differences. The results of the different loss functions are reported in Table 3. One can notice that all loss functions have gained fairness over our baselines (as reported in Table 2), while the permutation loss with the hidden representation $\mathbf{h}$, outperforms the others, and specifically Permutation/h. This implies that the hidden representation is important. In the Pokec datasets, the performance metrics of all losses, apart from the paired loss, are not impacted. We hypothesize this is caused due to its non-convexity in adversarial settings. Additionally, the paired loss demonstrates the same phenomena again: the lower the performance, the better the fairness. In the NBA dataset we do not see much difference between the loss functions. This can be explained due to the size of the graph. However, we do see that the permutation loss is the only one to improve fairness metrics while not hurting the performance metrics. Finally, we can notice that, paired loss functions (the permutation loss and the paired loss) perform better than the unpaired loss (apart from NBA, where the unpaired loss hurts the performance metrics). This can be explained by our paired problem, where we check for the difference between two scenarios of a node (real and fake). This illustrates the general importance of a paired loss function for paired problems.

6 Conclusions

In this work, we explored fairness in graphs. Unlike previous work which optimize the *statistical parity* (SP) fairness criterion, we present a method that learns to optimize *equalized odds* (EO). While SP promises equal chances between groups, it might cripple the utility of the prediction task as it does not give equalized opportunity as EO. We propose a method that trains a GNN model to neglect information regarding the sensitive attribute only with respect to its label. Our method pretrains a sampler to learn the distribution of the sensitive attributes of the nodes given their labels. We then continue training a GNN classifier while regularizing it with a discriminator that discriminates between true and sampled sensitive attributes using a novel loss function – the "permutation loss". This loss allows comparison of pairs. For the unique setup of adversarial learning over graphs, we show it brings performance gains both in utility and in EO. While this work uses the loss for the specific case of nodes in two scenarios: fake and true, this loss is general and can be used for any paired problem. For future work, we wish to test the novel loss over additional architectures and tasks. We draw the reader attention that the C2ST discriminator is the commonly used discriminator for many architectures that work over paired data, e.g. the pix2pix architecture. Replacing it with a paired discriminator might create a much powerful architecture.

We empirically show that our method outperforms baselines using fairness-performance metrics, over datasets with different attributes and sizes. To the best of our knowledge, we are the first to optimize GNNs for the EO criteria and hope it will serve as a beacon for works to come.

References

Barocas, S.; Hardt, M.; and Narayanan, A. 2019. *Fairness and Machine Learning*. fairmlbook.org. http://www.fairmlbook.org.

Belkin, M.; and Niyogi, P. 2001. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems*, volume 14, 585–591.

Bose, A.; and Hamilton, W. 2019. Compositional fairness constraints for graph embeddings. In *International Conference on Machine Learning*, 715–724. PMLR.

Buyl, M.; and De Bie, T. 2020. DeBayes: a Bayesian method for debiasing network embeddings. In *International Conference on Machine Learning*, 1220–1229. PMLR.

Cui, Z.; Henrickson, K.; Ke, R.; and Wang, Y. 2019. Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting. *IEEE Transactions on Intelligent Transportation Systems*.

Dai, E.; and Wang, S. 2021. Say No to the Discrimination: Learning Fair Graph Neural Networks with Limited Sensitive Attribute Information. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 680–688.

Dwork, C.; Hardt, M.; Pitassi, T.; Reingold, O.; and Zemel, R. 2012. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, 214–226.

Grover, A.; and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 855–864.

Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 1025–1035.

Hardt, M.; Price, E.; and Srebro, N. 2016. Equality of Opportunity in Supervised Learning. *Advances in Neural Information Processing Systems*, 29: 3315–3323.

Kang, B.; Lijffijt, J.; and Bie, T. D. 2019. Conditional Network Embeddings. *International Conference on Learning Representations*.

Kang, J.; He, J.; Maciejewski, R.; and Tong, H. 2020. InFoRM: Individual Fairness on Graph Mining. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 379–389.

Kingma, D. P.; and Ba, J. 2015. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. *International Conference on Learning Representations*.

Lopez-Paz, D.; and Oquab, M. 2017. Revisiting classifier two-sample tests. In *International Conference on Learning Representations*.

Obermeyer, Z.; Powers, B.; Vogeli, C.; and Mullainathan, S. 2019. Dissecting racial bias in an algorithm used to manage the health of populations. *Science*, 366(6464): 447–453.

Odén, A.; Wedel, H.; et al. 1975. Arguments for Fisher's permutation test. *The Annals of Statistics*, 3(2): 518–520.

Pedreshi, D.; Ruggieri, S.; and Turini, F. 2008. Discrimination-aware data mining. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 560–568.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 701–710.

Rahman, T. A.; Surma, B.; Backes, M.; and Zhang, Y. 2019. Fairwalk: Towards Fair Graph Embedding. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, 3289–3295.

Romano, Y.; Bates, S.; and Candès, E. J. 2020. Achieving Equalized Odds by Resampling Sensitive Attributes. *Advances in Neural Information Processing Systems*.

Roweis, S. T.; and Saul, L. K. 2000. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500): 2323–2326.

Singer, U.; Guy, I.; and Radinsky, K. 2019. Node Embedding over Temporal Graphs. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, 4605–4612. International Joint Conferences on Artificial Intelligence Organization.

Singer, U.; Radinsky, K.; and Horvitz, E. 2020. On biases of attention in scientific discovery. *Bioinformatics*. Btaa1036.

Takac, L.; and Zabovsky, M. 2012. Data analysis in public social networks. In *International scientific conference and international workshop present day trends of innovations*, volume 1.

Tenenbaum, J. B.; De Silva, V.; and Langford, J. C. 2000. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500): 2319–2323.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *International Conference on Learning Representations*.

Wang, H.; and Li, Z. 2017. Region Representation Learning via Mobility Flow. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 237–246.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Philip, S. Y. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*.

Yan, S.; Xiong, Y.; and Lin, D. 2018. Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Yan, S.; Xu, D.; Zhang, B.; Zhang, H.-J.; Yang, Q.; and Lin, S. 2006. Graph embedding and extensions: A general framework for dimensionality reduction. *IEEE transactions on pattern analysis and machine intelligence*, 29(1): 40–51.

Yu, B.; Yin, H.; and Zhu, Z. 2018. Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 3634–3640.

Zhang, B. H.; Lemoine, B.; and Mitchell, M. 2018. Mitigating unwanted biases with adversarial learning. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, 335–340.