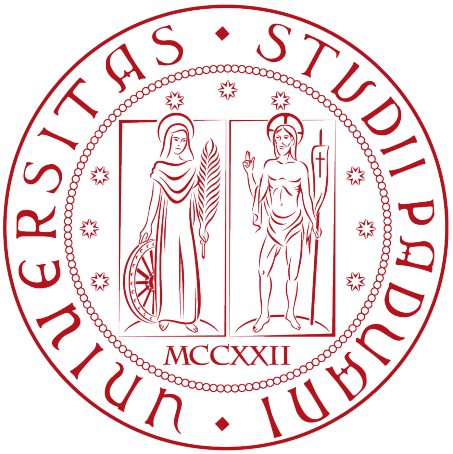
Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea Magistrale in Informatica



Enforcing fairness in graph representation learning

Tesi di Laurea Magistrale

Relatore

Prof.Nicolò Navarin

Laureando

Federico Caldart

Anno Accademico 2021-2022

Acknowledgements

Working on this Master's thesis have been a great experience. Writing this last document made me realize that every objective is achievable, provided enough commitment. However, I know that I would not be here without all the support I received in those last years.

I am deeply grateful to my supervisor Prof. Nicolò Navarin, that guided me in every phase of this work, motivating me and teaching me many things that not every professor would have been able to transmit; in this regard, I would like to underline how involving me in the writing of the paper "Biased Edge Dropout in NIFTY for Fair Graph Representation Learning" have been an incredible and profoundly instructive opportunity.

I would like to extend my sincere thanks to all my friends that believed in me and have been by my side in the last three years; a special mention must go to my group, "Trattore" that always made me feel proud of myself. Thanks to Alberto, Davide and Giovanni for the help and the wonderful moments shared in the last years, for me you've been more than classmates and found a place in my heart.

Finally, a special thanks to all my family, that let me continue my university career and believed in my potentialities. A special thanks to my brother Eugenio: maybe you do not know it, but your presence has always motivated me.

Padova, Luglio 2022 Federico Caldart

Abstract

As the representations output by Graph Neural Networks are increasingly employed in real-world applications, it becomes important to ensure that these representations are fair; graph embeddings can in fact encode potentially harmful social biases, such as the information that women are more likely to be nurses, and men more likely to be bankers. This work explores and enforces state of the art methods to mitigate this bias and produce fairer representations of real world graph data while maintaining a good classification accuracy.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Since their invention, programmable computers have been helping humans reduce effort and time in solving problems that are easily and formally translatable into an algorithm: a sequence of instructions to execute in order to transform a given input into an expected output. An example is sorting, where the input is a set of numbers and the output is their ordered list: there are various and different approaches to solve this task, but all of them can be precisely expressed as a succession of well-defined actions that will, in the end, produce the desired result.
However, many tasks are too complex to be formally and mathematically defined to be understandable by a machine, even if very intuitive to solve by humans. Let's take two opposite examples to make it more clear: a modern pocket calculator can easily solve any arithmetic operation because it can be defined as a fixed sequence of steps; on the other hand, how can a machine identify a crime by for example looking at a photo? If we are offered a picture of a person pointing a gun at a cashier, we would probably think that the person is trying to rob the content of the cash register, because in our head we associate the scene with a stealing attempt. However, if we think about it, it's not intuitive how to describe the process that our mind automatically performs when looking at the image; in fact, our ability comes from a continuous and subjective process of learning that made us able to recognize objects and situations.

Therefore, the key concept here is to come up with methods to teach a computer how to learn: a pocket calculator does not learn at all, in fact, every time we press the square root button the computation will be the same and it will always take the same time to be executed; a camera, to be able to correctly recognize a different crime, must learn to distinguish people, postures, weapons, places, etc. The approach must then be different: we have to allow computers to learn from experience, by providing many data that will make the machine understand the world in terms of concepts and their relations.

Luckily, with the advances in technology, we now have the ability to store and process large amounts of data that permitted the development of complex Machine Learning (ML) systems that can effectively achieve or even surpass human performance in many real-life tasks; as a consequence, the last decade has seen an escalation of diverse applications to which machine learning is applied: those models are nowadays used to filter loan applications, bail decisions, school admissions, and many more decisions. Then an important question arises: is it safe to let machines take

a particular outcome only basing their knowledge on past and already seen data? The answer is, unfortunately, that the high-performance level allows these models to also inherit *human biases* hidden in the data: generally speaking, in ML, bias is any disproportionate inclination towards or against an idea, that influences the behavior and results of an algorithm. Since data are created, refined, or generated from humans or from historical records that codify previous human decisions, beliefs, ideas, ways of thinking and prejudices can all be injected into the datasets, even unconsciously. As an example, think about a chatbot created to be free to learn from whoever interacts with it; even if it's programmed to be extremely polite, if a group of people starts to chat with racist or misogynistic language, it will learn a lot of dangerous and unethical concepts that will use to interact with other people. Another very popular example of racism in Artificial Intelligence is the issue with black people with face recognition: it has been demonstrated that those systems work better with white men, even if designed to recognize any face; that can happen, for example, because of unbalanced data where the majority of faces are of white males or because the team that developed the model is composed of all white men that unconsciously didn't take in considerations physiognomic characteristic of females and black people. Or think about a machine designed to support judges in court; imagine that all historical sentences starting from the early 1900s are fed to the algorithm to learn from: the machine could learn that Jews or black people should be almost always judged guilty, because of a lot of old examples with that result.
It is then clear that when the decisions of these models may affect people's lives we cannot allow these biased behaviors, because they would produce unwanted and unethical discrimination. In particular, we would like our models to be fair, namely to treat equally different subgroups of the population based on characteristics such as gender or ethnicity (like in the previous examples), referred to as sensitive attributes.

The problem is even more challenging when the input data is complex (e.g., graphs) and back-box models such as Graph Neural Networks (GNNs) need to be employed to achieve satisfactory performance. GNNs, as the name suggests, are algorithms designed to work with graphs; a graph is a complex structure, useful to represent all those data that can be viewed as network entities connected by some sort of relationships, depending on the context. the two fundamental elements of a graph are nodes, i.e. individuals, items, and entities for which we have a set of characteristics, referred to as *attributes*, and edges, i.e. connections between nodes. A straightforward example is social networks, that can be modeled as graphs where nodes represent people, attributes are all the characteristics of a person, like her age, gender, country, etc and edges are the friendship relationships between those people.
GNNs in particular learn by aggregating information from neighbor nodes: in the example of social networks, nodes tend to connect to other nodes with similar attributes, leading to denser connectivity between nodes with the same sensitive attributes (e.g., gender) Thus, a model that learns from those data may highly correlate with people because of shared sensitive attributes; this can cause discrimination even when the sensitive attributes are not directly used or increase it with respect to other models that do not use neighbors' information.

After a review of state-of-the-art models, we noted that different methods have been proposed in the literature to learn fair graph node representations, mainly including additional terms in the loss function to account for some definition of fairness, or perturbing the graph topology in a biased way. As a representative of the first family

of methods, NIFTY [1] proposes that some of the additional terms can be based on perturbing the input data such as node attributes (including the sensitive attribute) or graph topology. The latter methods are based on the intuition that GNNs tend to smooth the learned representations of connected nodes, and if the graph shows high homophily with respect to the sensitive attribute (i.e., nodes with the same value for the sensitive attribute tend to be connected), the GNN may introduce inequalities in the learned representations, because aggregating information of neighbor nodes may induce the model into learning that people that share the same sensitive attributes are always similar, even if they share very few other characteristics.
This thesis proposes to unify the two families of approaches, using NIFTY as a baseline. We first concentrate on modifying the approach used to perturbate the graph topology by following and adapting the intuition given by FAIRDROP [70]. Then we focused our attention on node attributes perturbation: after running experiments with and without sensitive attributes, we noted, as expected, that removing them wasn't enough to sensibly reduce the unfairness in the solutions found by the models, even when not considering arcs, so not taking in account the information provided by the graph structure. Then, following the idea of FairRF [79], we modified NIFTY's attribute perturbation by considering the correlations with the sensitive one. Experimental results on four real-world datasets show that our proposal can improve different fairness metrics compared to the original NIFTY formulation while maintaining the same computational complexity and the same level of predictive performance.

This thesis is structured as follows. Chapter 2 describes the background needed to better understand the concepts covered in the following ones; in particular, it gives an introduction to machine learning, deep learning and Neural Networks (NNs), graphs, and fairness issue with motivations, followed by a more detailed description of related works. Chapter 3 describes initial and state-of-the-art methods used to learn on graphs. Chapter 4 reports our contribution in terms of modified models and chapter 5 reports experimental results with relative discussion. Finally, chapter 6 sums up the contribution of this work and proposes future works to possibly extend the results obtained by us.

Chapter 2

Background

This chapter will introduce and describe all the knowledge needed to fully understand what this thesis is about. It starts from a general definition of Machine Learning to the more specific setting of learning on graphs, which is one of the main topics of this thesis, and it ends with a section entirely dedicated to fairness in machine learning, explaining what motivated us to focus on that subject and its importance in the field of Artificial Intelligence.

2.1 Machine Learning

Artificial intelligence (AI) is a thriving field with many practical applications and active research topics whose main goal is to create intelligent software to automate time and mind-consuming tasks; its straight-forward application is to solve problems intellectually difficult for humans, but easily definable in a formal mathematical way that is understandable by computers. Instead, if we think about tasks that our mind is able to compute with ease, like for example recognizing a face, classifying the topic of a document, or identifying entities (places, names, etc) in a phrase or an image, it's challenging to understand how to formalize the problem in a way that is understandable from a computer, since we learned how to do it by growing and making experience.
The real challenge then comes when it's not possible to exactly formalize the problem, when there is a certain level of uncertainness on input or output and when the solution is too complex or too inefficient. One of the first definitions of machine learning was given by Arthur Samuel in 1959 :

"Machine learning is the field of study that gives computers the ability to learn without being explicitly programmed"

In fact, machine learning allows us to tackle those issues by learning from experience and understanding notions by their relation to simpler concepts with a hierarchical view; a machine learning algorithm is able to adapt to the environment in which is implemented while improving its performance w.r.t. a particular task. Data have a fundamental importance in the learning process: the task is usually defined in terms of how the system should process an example; an example is a set of features and is typically represented as a vector $x \in \mathbb{R}^n$ where each entry x_i of the vector is a feature.

For example, the features of a node representing a user in a graph are all the attributes associated with that user (e.g. gender, age, education, location, etc) and the relations it has with other nodes in the graph (i.e., edges).

There are many different tasks that can be solved with machine learning: classification, regression, transcription, machine translation, anomaly detection, sampling, denoising, image or example generation, etc. This work will focus on classification and in particular on **binary node classification**; more details will be given in section 2.3. Any task for which machine learning is applied follows a fundamental assumption on data:

"There is a stochastic process which explains observed data. We do not know the details about it but we know it is there!"

Hence, the common goal of any machine learning system is to build good approximations of this process.

Data and task also influence what kind of experience an ML algorithm is allowed to have during the learning process; there are two main branches of machine learning:

* **Supervised learning**: data are presented as a set of (x, y) pairs as **training set**, where x is a sample with its features and y is the corrected output, also called **target**. A supervised learning algorithm exploits the knowledge about the target to infer a function able to predict y given a new x, typically estimating the probability distribution $P(x|y)$. Depending on the target possible values, a supervised learning system can have different kinds of outputs: binary, multiclass, multilabel, and regression.
* **Unsupervised learning**: data in the training set are presented without labels, so there is no target y. In this case, the goal is to identify useful patterns and properties of the structure of the dataset, so as to infer the prior probability $P(X)$. There is no instructor or teacher, so the model must learn to make sense out of the data only; an example is clustering, where the objective is to group examples into different sets, based on the values of their features.

However, the two settings are not completely separated; there is in fact a variant called **Semi-supervised learning**, where some examples include a known target while other don't. In this case the dataset X can be divided into two parts: the points $X_l := (x_1, ..., x_l)$ for which labels $Y_l := (y_1, ..., y_l)$ are provided, and the points $X_u := (x_{l+1}, ..., x_u)$ for which labels are unknown.

It is not straightforward to define which setting this work fits in because of the atypical nature of graphs and node classification, which blurs the boundaries between standard approaches; more details and motivations will be given in 2.3.2, but as for now we can consider the supervised learning setting.

2.1.1 Supervised learning for classification

This section defines more specifically the supervised setting for the classification task. The goal is to learn a mapping from inputs x to outputs y, where $y \in 1, ..., c$, where c

is the number of *classes*: when $c = 2$ then the task is called *binary classification* and we assume $y \in 0, 1$; when $c > 2$ then the task is called *multiclass classification*. There are also cases in which a single example could be classified with more than one class and we call it *multi-label classification*, or *multiple output model* if viewed as predicting multiple paired binary class labels: for example, one can be tall or short and female or male.
The problem can be formalized as approximating a function $\hat{f}(x) = \hat{y}$ to make predictions on inputs not seen before (**test set**), the idea comes from the assumption that $f(x) = y$ for an ideal and unknown f.
From a probabilistic point of view, this is the same as assuming that the set of pairs (x, y) are generated according to a probabilist function $P(x, y) = P(x)P(y|x)$, so the goal is to estimate the probability $P(y|x)$: in the case of binary classification, it is then sufficient to consider $P(y = 1|x)$, since $P(y = 1|x) + P(y = 0|x) = 1$.
To consider the model able to generalize for unseen data and to describe the data generating process with a probability distribution over a single example, **i.i.d assumptions** are made: examples in the dataset are statistically independent from each other and are identically distributed. In section 2.3 we will see why this assumption is not entirely true in the context of learning on graphs for the reasons already introduced at the end of the previous section.

2.1.2 Artificial Neural Networks

Artificial Neural Networks (ANNs) or Neural Networks (NNs) are among the most effective machine learning methods currently known. Historically, two were the motivations that inspired the research in NNs: the first, biological, was the purpose of reproducing human brain behavior, by creating very complex webs of interconnected neurons; the second was the attempt to extract the fundamental principle of calculus and the goal of obtaining highly effective machine learning algorithms, independently from the biological correlation.
An ANN is not in itself an algorithm, but rather a system consisting of interconnected simple units that compute *nonlinear (numerical) functions*, where each unit takes a number of real-valued inputs (possibly the output of other units) and produces a single real-valued output. Those single units are called *artificial neurons* since they vaguely model neurons in a biological brain: connections between units work like real synapses, sending signals between neurons; adjustable **weights** are associated with connections among units and are modified in the training process: the weight increases or decreases the signal strength of the connection.
The first and most common artificial neuron is the *perceptron*, invented in 1943 by McCulloch and Pitts [50] and implemented for the first time in 1958 at the Cornell Aeronautical Laboratory by Frank Rosenblatt [63]. The perceptron is a learning algorithm for binary classification, that is, a function that maps its input $x \in \mathbb{R}^n$ in an output value $o(x)$, with $o(x) = 1$ if the result is greater than some threshold and -1 otherwise. More precisely, o is defined as:

$$o = f(w^T x + b) \tag{2.1}$$

Where $\mathbf{w} \in \mathbb{R}^n$ is the vector of learnable weights, b is called *bias*, is independent from the input and can be used to translate the decision boundary; it can however be incorporated into the vector $\mathbf{w}$, consequently adding an element with value 1 in the input vector $\mathbf{x}$. The threshold is determined by a function $f(\cdot)$, that can be defined as:

$$f(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.2}$$

This function corresponds to the function $sign(\cdot)$, which simply returns the sign of its input and the model is represented in figure 2.1. In this setting, a single perceptron can then be used to represent many boolean functions, but only if the examples are **linearly separable**. In fact, non-linear element-wise transformation are mostly used as activation functions (sigmoid, ReLu, etc).

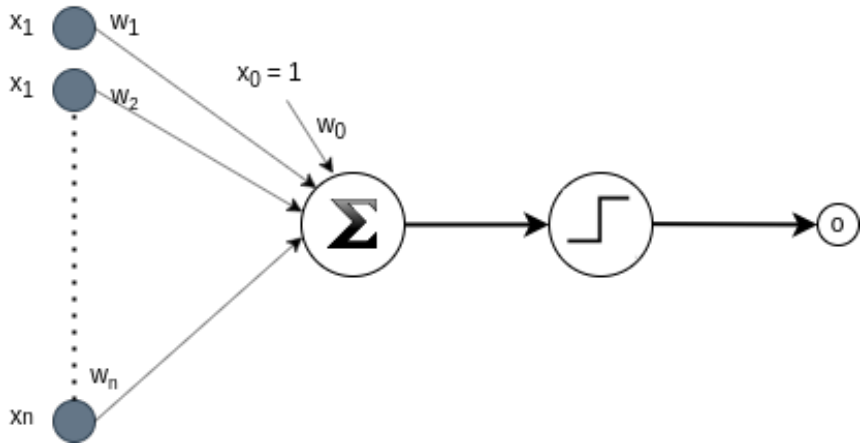


Figure 2.1: Perceptron with linear activation function $sign(\cdot)$

Furthermore, a single perceptron has not enough capacity to represent complex functions, since it can only express linear decision surfaces. The solution to this limitation is the use of **feedforward neural networks** that aggregate multiple neurons in groups called *layers*, with each element linked to every element of the previous layers with an adjustable weight to form a network. The idea is that by stacking multiple layers we can compose and compute different functions, for example, a network with 4 layers can be seen as:

$$f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x))) \tag{2.3}$$

In the equation 2.3, $f^{(1)}$ and $f^{(2)}$ are the **hidden layers**, that represent internal variables and $f^{(3)}$ is the **output layer**. We can derive from this formulation that each layer is a function of the preceding one, that is fed into it, as shown in figure 2.2; to define the network the same way as we did for the single perceptron, we use the notation $h^{(i)}$ to identify the i-th layer output and $g^{(i)}$ to identify the activation function for the i-th layer:

$$\begin{aligned} h^{(1)} &= g^{(1)}(W^{(1)T}x + b^{(1)}), \\ h^{(i)} &= g^{(i)}(W^{(i)T}h^{(i-1)} + b^{(i)}) \end{aligned} \tag{2.4}$$

This architecture explains why they are called *forward* neural networks: information flows through the function being evaluated from x, through the intermediate computations used to define h, and finally to the output; There are no feedback connections

in which outputs of the model are fed back into itself. The number of layers (i.e., the depth of the network) and hidden units (i.e., the width of the layers) are two **hyperparameter** that do not depend directly on the problem but must be tuned to get the best results for the selected performance measure. Deeper networks tend to have fewer parameters and to generalize better but are also harder to train, but work better than shallow ones only if the function to learn can be expressed as a composition of simpler functions. Moreover, the **universal approximation theorem** [28] says that a feedforward network with (at least) one hidden layer and a linear output can approximate any continuous function with a tolerance up to a small value ϵ at will; in the worst case, an exponential number of hidden units is needed, but deeper models may reduce that number.

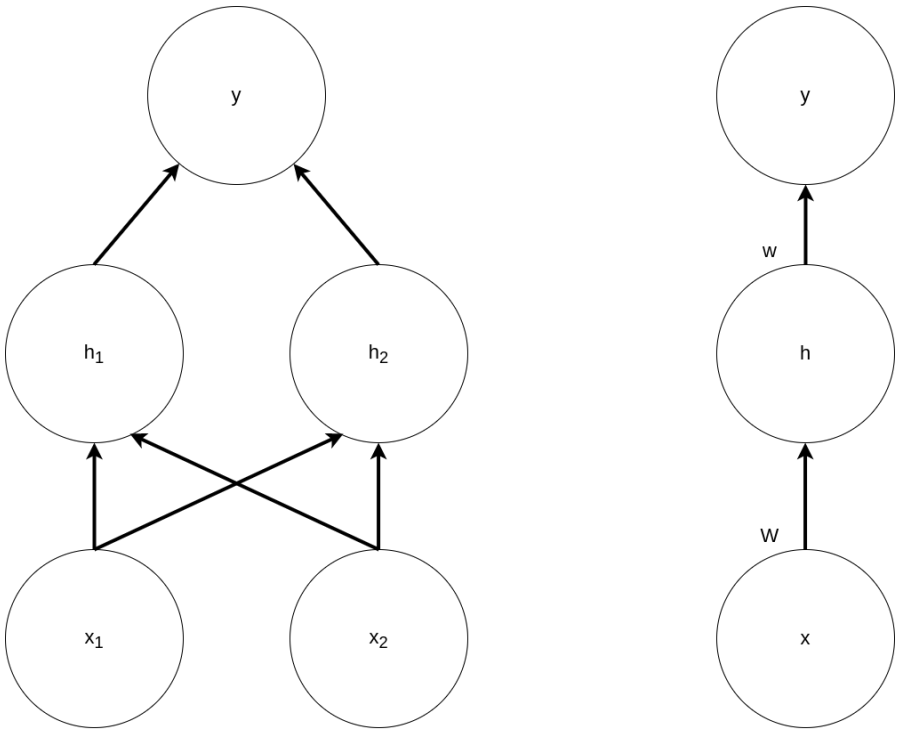


Figure 2.2: Two different representations for the same feedforward neural network.

2.2 Graphs

Graphs are a ubiquitous data structure for describing complex systems. In the most general view, a graph is simply a collection of objects (i.e., nodes), along with a set of relations (i.e., edges) between pairs of these objects.
An example is encoding a social network, where nodes can represent individual users and edges represents the friendship relationship between them (2.3); or in e-commence, a graph can define users and products as nodes and the interactions between them as edges; another popular example is in the biological domain where nodes can represent proteins, and edges represent various biological interactions between them.

Graphs are a powerful formalism because they do not only focus on the properties of

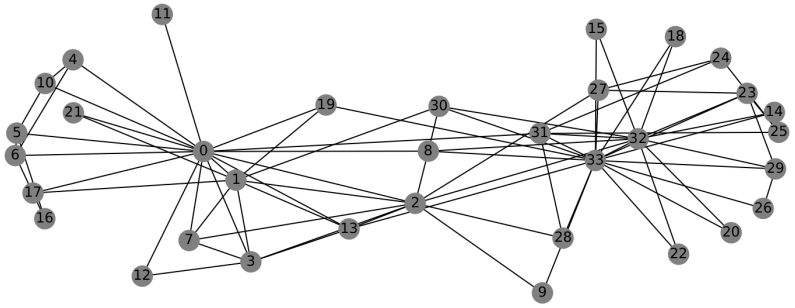


Figure 2.3: Zachary Karate Club Network - represents the friendship relationships between members of a karate club

individual examples (i.e., node attributes) like standard machine learning approaches, but also on the relationships between those examples. Another important characteristic is that their usage is not restricted to a close set of domains, since as said before in several settings it is natural to represent data as graphs, e.g. with social networks, interactions between drugs and proteins, interactions between atoms in a molecule, or the connections between terminals in a telecommunications network.

Graphs also offer a mathematical foundation that we can use to analyze, understand, and learn from all those real-world complex systems. With the advent of large-scale social networking platforms, databases of molecule graph structures, billions of interconnected devices, etc, there is no shortage of meaningful graph data for researchers to analyze.

2.2.1 Formal definition and notation

Before discussing how machine and deep learning can be applied to graphs, it is necessary to provide some formal definitions.

Definition 2.1. (Graph) A graph $G = \{V, E, X, s\}$ is defined by a set of nodes $V = \{v_0, ..., v_{n-1}\}$ and a set of edges $E \subseteq V \times V$ between these nodes, $X \in \mathbb{R}^{n \times d}$ is the matrix of non-sensitive attributes, and $s \in \{0, 1\}^n$ is the vector associating a value for the binary (for sake of simplicity) sensitive feature of each node. We denote an edge going from node $u \in$ in V to node $v \in V$ as $(u, v) \in E$. We define $A \in \mathbb{R}^{n \times n}$ as the adjacency matrix of the graph: we order the nodes in the graph so that every node indexes a particular row and column in the adjacency matrix, so its elements $a_{i,j} = 1 \iff (v_i, v_j) \in E$. With $\mathcal{N}(v)$ we denote the set of nodes adjacent to node v. Let also $\tilde{D} \in \mathbb{R}^{n \times n}$ be the diagonal degree matrix where $d_{ii} = \sum_j a_{ij}$, and $\mathbf{L}$ the normalized graph Laplacian defined by $\mathbf{L} = \mathbf{I} - \tilde{D}^{-\frac{1}{2}}\mathbf{A}\tilde{\mathbf{D}}^{-\frac{1}{2}}$, where $\mathbf{I}$ is the identity matrix.

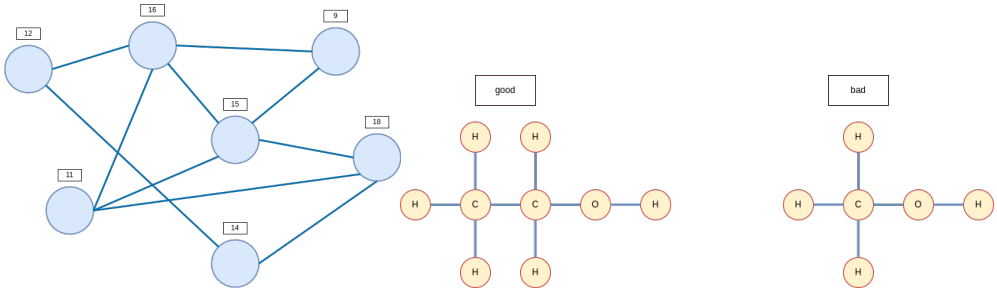
In our case edges are all undirected, i.e., $(v_i, v_j) \in E \iff (v_j, v_i) \in E$, so A will be a symmetric matrix, but a graph can also be directed (i.e., edge direction matters); in this case A will not necessarily be symmetric. Some graphs can also have weighted edges, where the entries in the adjacency matrix are arbitrary real-values rather than $\{0, 1\}$. For instance, a weighted edge in a protein-protein interaction graph might

indicate the strength of the association between two proteins. This work only focuses on unweighted edges setting.

2.3 Learning on graphs

Taking again the example of social network data, the single data "points" (i.e., users) are closely interrelated, and not explicitly representing such dependencies would inevitably lead to an information loss. Thus, it clearly is important to come up with methods to efficiently exploit the structural information.
Machine learning is a problem-driven discipline, where models are usually categorized according to the type of task they seek to solve, as seen in section 2.1. Working with graphs makes no difference, but the standard categories, i.e.,*supervised* and *unsupervised* learning are not exactly the most suited when it comes to graph data. There are two main tasks that we may want to pursue (see figure 2.4):

- Predict some property over entire graphs (e.g. graph classification); in this case, supervised learning is ok;
- Predict property over the nodes and/or edges of a single, large graph (e.g. node classification, link prediction);



(a) Node-level predictions. **(b)** Graph-level predictions.

Figure 2.4: Two examples of tasks on graphs: (a) node regression; (b) graph classification.

In any case, learning on graphs can be difficult in general for two main reasons:

1. **Graph Isomorphism**: a graph $G_1 = (V_1, E_1)$ is isomorphic to $G_2 = (V_2, E_2)$ if there exists a bijiective mapping $f : V_1 \times V_2$ such that $\forall (v_1, v_1^{'}) \in E_1 \iff (f(v_1), f(v_1^{'})) \in E_2$.
2. **Subgraph Isomorphism**: a subgraph $G_2 = (V_2, E_2)$ of $G_1 = (V_1, E_1)$ is a graph for which $V_2 \subseteq V_1$, $E_2 \subseteq E_1 \cap (V_2 \times V_2)$.

The model learning result must in fact be **invariant** to the way the graph is represented (i.e., in which order and how nodes and edges are presented); in fact, graphs that are *isomorphic* should induce the same function over their nodes.

2.3.1 Graph representation learning

When learning on graphs the general and main goal is to obtain a sufficiently rich *embedding* of the graph and of each node that is able to capture attribute information

as well as structural properties of the input data. In other words, we want to project (or encode) nodes into a latent space where relationships in this latent space correspond to relationships (e.g., edges) in the original graph [27].
While many methods have been studied and implemented in literature, we can unify them all in an *encoder-decoder* framework [23]; in this framework, the encoder maps each node to a low-dimensional embedding, and a decoder decodes structural information about the graph from the encoded embedding: if a model is able to decode high-dimensional graph information from the low-dimensional representation, then these representations should contain all the necessary information needed for the more specific *downstream* task. As an example, a decoder might predict the existence of edges between nodes or nodes labels, taking as input their embeddings (see figure 2.5).

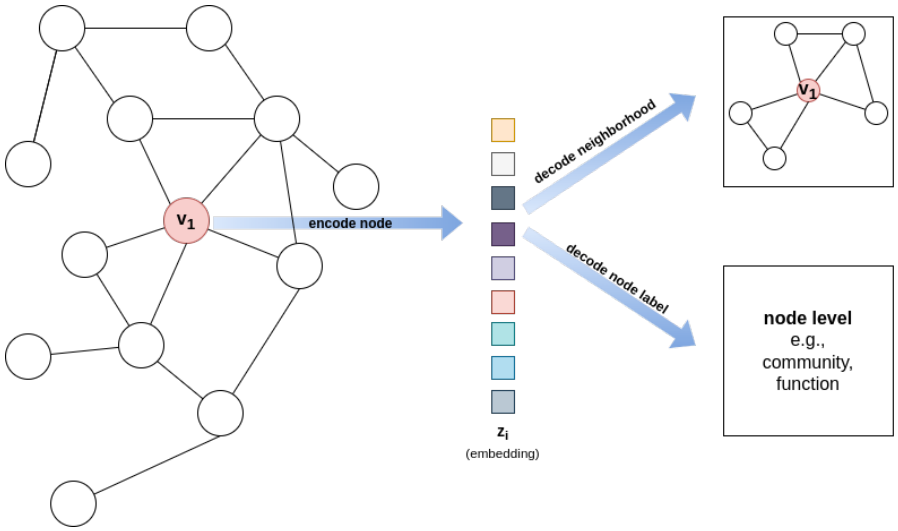


Figure 2.5: Illustration of the encoding-decoding process and downstream tasks.

This work will focus on deep NN-based models that provide an end-to-end solution to both representation learning and different downstream tasks, but for the sake of completeness we cite two traditional families of approaches used in literature:

* *Factorization-based* approaches follow the idea that decoding local graph structure from a node's embedding can be the same as reconstructing the entries in the adjacency matrix; matrix factorization is used to learn a low-dimensional approximation of node-node similarity that captures some user-defined similarity notion. This can be done with the Laplacian eigenmaps technique [4] or with inner-product methods [8], [2];

* *Random walk* approaches adapt inner-product technique to use *stochastic* measures of similarity [31]

A popular example of *random walk* approach is *node2vec* [20]. The algorithm transposes the procedure used in *word2vec* in the context of graphs; *word2vec* is an algorithm used to perform word embedding, i.e., to transform words in a numerical representation (vector). The main feature of *word2vec* is its ability to group together vectors of similar words by adopting the *skip-gram* architecture, It is implemented as a

NN trained on pairs of consecutive words selected over a rolling window in a text; once trained, the network will yield a probability that indicates the similarity between a target word and other words in the corpus, i.e., the probability of being near in the text.

Graphs are however not linear structures, so the sliding window technique cannot be applied. *Node2vec* proposes a procedure to sample neighborhoods of a given source node v that are not restricted to just immediate neighbors, by employing a biased random walk procedure to explore neighborhoods. Formally, a random walk starting from a node v is a list of nodes of length l, where each node v_i is generated by the following distribution:

$$P(v_i = a | v_{i-1} = b) = \begin{cases} \pi_{b,a} & \text{if } (b,a) \in E \\ 0 & otherwise \end{cases} \tag{2.5}$$

Where $\pi_{b,a}$ is the unnormalized transition probability between nodes b and a and Z is the normalizing costant.

Node2vec modifies $\pi_{b,a}$ by introducing two parameters p and q that control how fast the walk explores and leaves the neighborhood; more specifically, p controls the likelihood of immediately revisiting a node in the walk, while q controls how the procedure is inclined on visiting further nodes. Formally, the unnormalized transition probability is set to $\pi_{b,a} = \alpha_{pq}(t,a) * w_{ba}$, where:

$$\alpha_{pq}(t,a) = \begin{cases} \frac{1}{p} & \text{if } d_{ta} = 0 \\ 1 & \text{if } d_{ta} = 1 \\ \frac{1}{q} & \text{if } d_{ta} = 2 \end{cases} \tag{2.6}$$

The algorithm uses this procedure to extract a set of random walks from the input graph, that are then fed into the skip-gram model to generate the embeddings.

However, traditional methods do not leverage node features in the encoding process, while many graphs do have informative features associated with the nodes; moreover, those methods are inherently *transductive*, i.e., they can only generate embeddings for nodes observed during the training process and cannot generalize for unseen nodes.

2.3.2 Node classification

This work is defined in the context of the node classification task, so it is important to reserve a section for it.

Let's start with an example. Suppose we are given the task from an IT company to identify suitable candidates for a machine learning job; we are given a large social network dataset with millions of users, but we know that most of these users are actually web developers; identify those users would greatly reduce the candidate search process. Manually examining every user to determine if they are ML specialists would be prohibitively expensive, so ideally we would like to have a model that could classify users as web developers (or not) given only a small number of manually labeled examples.

This is a classic example of node classification, where the goal is to predict the label y associated with all the nodes $v \in V$, when we are only given the true labels on a training set of nodes $V_{train} \subset V$. Node classification seems to be the most popular machine learning task on graph data: examples of node classification beyond social networks

include classifying the function of proteins in the interactome and classifying the topic of research papers based on hyperlink or citation graphs. Often, like in this work, we can assume that we have label information only for a very small subset of the nodes in a single graph (like in the example). However, there are also instances of node classification that involve many labeled nodes and/or that require generalization across disconnected graphs (e.g., classifying the function of proteins in the interactomes of different species).

Node classification appears then to be a straightforward variation of standard supervised classification, where the only thing that changes is the type of data but, as mentioned above, some considerations have to be made before categorizing some graph learning methods in a standard setting.

The most important difference is that the nodes in a graph are not independent and identically distributed (i.i.d.). In fact, when building standard supervised machine learning models, we assume that each example in the data is statistically independent of all the others; otherwise, we might need to model the dependencies between all our input points. We also assume that the examples are identically distributed; otherwise, we have no way of guaranteeing that our model will generalize to new inputs. Node classification completely breaks this i.i.d. assumption, because we are instead modeling an **interconnected** set of nodes. In fact, if we do not consider the relationships (i.e., edges) between nodes, we are missing some information that can be crucial to learn the correct embeddings and predict the correct labels; moreover, sometimes relations information can be more informative than nodes' features: for example by exploiting homophily, which is the tendency for nodes to share attributes with their neighbors in the graph, we can more easily identify subgroups of users; as a concrete example, in social networks, people tend to form friendships with others who share the same interests or demographics: based on the notion of homophily we can build machine learning models that try to assign similar labels to neighboring nodes in a graph. Same thing for heterophily, which presumes that nodes will be connected to nodes with different labels.

For this reason, node classification problems are often referred to as **semi-supervised**, because when we are training node classification models, we usually have access to the full graph, including all the unlabeled nodes. So we can still use information about the test nodes to improve our model during training, also because test and train nodes can potentially be connected, directly (they are neighbors) or indirectly (they share some neighbors). We will now describe the process of applying GNNs in the node classification task and will return to the question at the end of the section.

The standard way to apply GNNs to a node classification task is to train GNNs in a fully-supervised setting, where we define the loss using a softmax classification function and negative log-likelihood loss:

$$\mathcal{L} = \sum_{v \in V_{train}} -log(softmax(z_v, y_v)) \tag{2.7}$$

Where we assume $y_v \in \mathbb{Z}^C$ is a one-hot vector indicating the class of the training node v; with $softmax(z_v, y_v)$ (remember that z_v is the output of the last layer) we denote the predicted probability that the node belongs to the class y_v, computed using the softmax function.

While training a GNN, we know that we have for sure the set of training nodes V_{train}, that is included in the message passing as well as in the computation of the

loss, as seen in 2.7. However, we can also have another set of nodes that we can identify as V_{trans} (from *transductives*): for these nodes, the model does not know the ground-truth labels, so they are not used in the loss computation, but they will be involved in the message passing process along with their edges. So the GNN will produce their hidden representations, but their final layer embedding will not be used for the loss computation. Node classification with GNN can therefore be called *semi-supervised* when V_{trans} nodes are used to test the solution because they have also been observed during training.

2.3.3 Graph Neural Networks

Graph Neural Networks (GNNs) are state-of-the-art methods for learning on graphs. The main feature of a GNN is the use of a form of *neural message passing* in which messages are exchanged between nodes and updated using neural networks. In this framework, during each message-passing iteration, a hidden representation h_v^i corresponding to each node $v \in V$ is updated including information from its neighborhood $\mathcal{N}(v)$.

This process can be generally specified by trainable operators MSG, AGG, UPD [1] and a typical layer is given by:

$$h_v^k = UPD(AGG(MSG(h_v^{k-1}, h_{v_j}^{k-1})|v_j \in \mathcal{N}_v), h_v^{k-1}) \tag{2.8}$$

In eq. 2.8, we can consider MSG as the message-passing operator between nodes v and v_j, AGG (typically a fully-connected layer) as the aggregation operator where messages from $\mathcal{N}_v$ are aggregated, and finally, UPD (typically a non-linear activation function) operator combines the aggregated message of neighbor nodes with the hidden representation of the node produced at the previous layer to construct a new updated representation. After running K iterations of the GNN message passing, we can use the output of the final layer to define the embeddings for each node:

$$z_v = h_v^{(K)}, \forall v \in V \tag{2.9}$$

See figure 2.6 for a high-level illustration of the process.
The basic intuition behind the GNN message-passing framework is that after the first iteration (k = 1), every node embedding contains information from its 1-hop neighborhood, i.e., every node embedding contains information about the features of its immediate graph neighbors, which can be reached by a path of length 1 in the graph; after the second iteration (k = 2) every node embedding contains information from its 2-hop neighborhood; in general, after k iterations, every node embedding contains information about its k-hop neighborhood.
The information encoded in this way can have two forms: **structural** and **feature-based**. Examples: for the former, after k iterations of GNN (k) message passing, the embedding h_v of node v might encode information about the degrees of all the nodes in v's k-hop neighborhood; for the latter, after k iterations of GNN message passing, the embeddings for each node also encode information about all the *features* in their k-hop neighborhood.
Feature-based information propagation connects to the general idea behind convolution on graphs, which is parallel with images convolution; in fact, in Convolutional NN for images, for each entry in the hidden representation, the computed representation depends on the corresponding input pixel and on the neighbors one [64]. However,

whereas CNNs aggregate feature information from spatially-defined patches in an image (i.e., defined by a filter), GNNs aggregate information based on local graph neighborhoods; moreover, CNNs can only operate on regular Euclidean data like images (2D grids) and texts (1D sequences) while graphs are non-euclidean data and it is hard to define localized convolutional filters, which hinders the transformation of CNN from Euclidean domain to non-Euclidean domain.

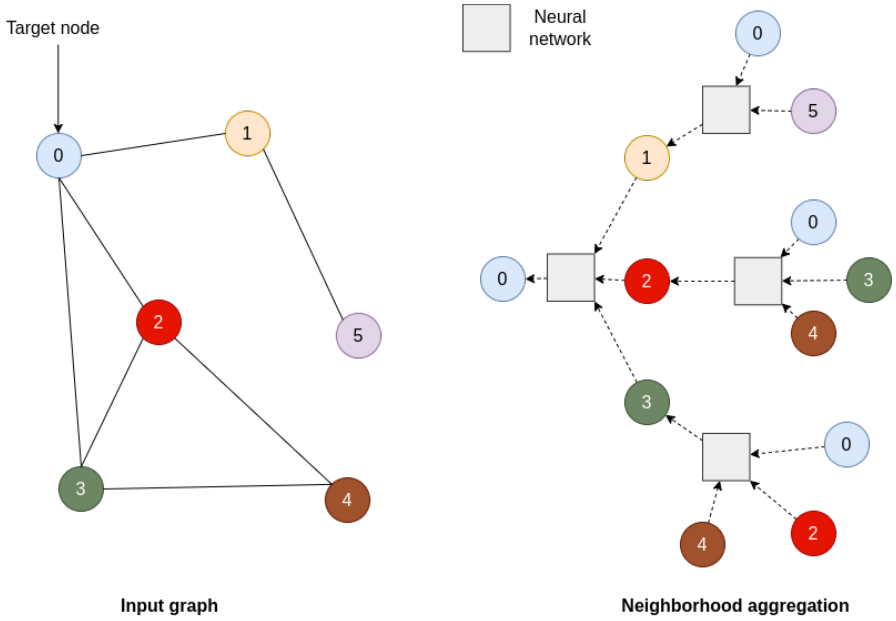


Figure 2.6: Message passing process. Given an input graph (a), GNN predicts the label of the target node (e.g., the blue node) by aggregating the information from neighboring nodes (b).

Early models and basic GNN definition

This section briefly reviews the history and early models of Graph Neural Network, along with the first notion of GNN outlined in the literature.
In 1997, Sperduti et al. [69] first applied neural networks to structured data, including directed acyclic graphs, proposing to generalize structures of a recurrent neuron able to build a map from a domain of structures to the set of reals; thanks to weights sharing, the same set of neurons is applied to each node in the graph, obtaining a representation that is based not only on the features on the node but also on the representations generated for its neighbors. Then, the first notion of GNNs was introduced by Gori et al [19] and further elaborated by Scarselli et al [66]. In the same year, Micheli [53] proposed the Neural Network for Graphs model. The formal definition is the following:

$$h_v^k = \sigma(W_{self}^{(k)} h_v^{(k-1)} + W_{neigh}^{(k)} \sum_{v_i \in \mathcal{N}(u)} h_v^{(k-1)} + b^{(k)}) \tag{2.10}$$

Where $W_{self}^{(k)}$, $W_{weight}^{(k)} \in \mathbb{R}^{d^{(k)} \times d^{(k-1)}}$ are trainable weight matrices and σ is an element-wise non linear transformation; we first sum the contribution given by the neighbors nodes, then we combine this information with the node's previous embedding

using a linear combination and finally we apply the non-linear function.
The same definition can be rewritten in terms of the previously defined operators (superscript omitted for notational brevity):

$$UPD(h_v) = \sigma(W_{self}h_v + W_{neigh}AGG(M_{\mathcal{N}(u)})) \tag{2.11}$$

with $M_{\mathcal{N}(u)}$ denoting the message passing operation:

$$M_{\mathcal{N}(u)} = \{MSG(h_v, h_{v_i})|v_i \in \mathcal{N}(v)\} \tag{2.12}$$

Those early and pioneer works have been grouped under the name *Recurrent Graph Neural Networks (RecGNNs)* [75] since they exploit recurrent neural architectures.
Several years after, the approach followed by Micheli was independently proposed in [13] under the name *graph convolution*. Before going into detail and describe some of the most widely studied and adopted graph convolutions in literature, we briefly introduce the derivation of spectral-based approaches, since many methods are applied in the graph spectral domain instead of defining convolutions directly on the graph topology.
Spectral approaches are theoretically based on graph signal processing [67],[9]. Let us fix a graph G. Let $x : V \to \mathbb{R}$ be a signal on the nodes V of the graph G, i.e. a function that associates a real value to each node of V. In spectral methods, a graph signal x is first transformed to the spectral domain by the graph Fourier transform $\mathcal{F}$, then the convolution operation is conducted. After the convolution, the resulted signal is transformed back using the inverse graph Fourier transform $\mathcal{F}^{-1}$. These transforms are defined as:

$$\mathcal{F}(x) = U^T x \tag{2.13}$$

$$\mathcal{F}^{-1}(x) = Ux \tag{2.14}$$

Where U is the matrix of eigenvectors of the normalized graph Laplacian L (see definition 2.1); the normalized graph Laplacian is real symmetric positive semidefinite, so it can be factorized as $L = U\Lambda U^T$, with Λ a diagonal matrix of the eigenvalues. The graph convolution of the input signal x with a filter $f \in \mathbb{R}^n$ is defined as:

$$f *_G x = U(\hat{f} \odot \hat{x}) = U((U^T f) \odot (U^T x)) \tag{2.15}$$

Where $\odot$ denotes the component-wise product. With some reformulations, not reported on this work but visible in , we obtain:

$$f *_G x = U\hat{F}U^T x \tag{2.16}$$

Where $\hat{F} = diag(\hat{f})$. We can now design the diagonal matrix $\hat{F}$ and thus the spectral filter f. One way is by using polynomial parametrization on powers of the spectral matrix Δ [55]. What we obtain then is a filter with $k+1$ $\{\Theta_0, ..., \Theta_k\}$ learnable parameters that can be formulated directly on the graph domain:

$$f *_G x = \sum_{i=0}^{k} \Theta_i L^i x \tag{2.17}$$

Spectral-based approaches follow this definition. The key difference between them lies in the choice of the parametrization of the polynomial filter.

Graph Convolutional Networks

Graph Convolutional Networks (GCNs) [41] modifies the approximation by Chebyshev polynomials of the diagonal matrix of eigenvalues presented in [13], with a first-order approximation and restraining the number of learnable parameters assuming $\Theta = \Theta_0 = \Theta_1$ to avoid *overfitting*. The convolution operator is then defined as:

$$f *_G x = \Theta(I_n + D^{-\frac{1}{2}} A D^{-\frac{1}{2}})x = \Theta(2I_n - L)x \tag{2.18}$$

Using $\Theta(I_n + D^{-\frac{1}{2}} A D^{-\frac{1}{2}})$ is numerically instable; to address this problem, GCN adopts a normalization trick to replace it with $\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$, with $\tilde{A} = A + I_n$ and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. Applying this convolution operator to a multivariate signal $X \in \mathbb{R}^{n \times r}$ and using m filters, a single graph convolutional layer is defined as:

$$H = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} X \Theta. \tag{2.19}$$

To obtain a GCN, several graph convolutional layer are stacked and interleaved by a non linear activation function.
GCN is a spectral-based method, but can also be interpreted as a spatial-based method; from that perspective, it can be considered as aggregating feature information from a node's neighborhood:

$$h_v = \Theta^T \Big(\sum_{v_i \in \{\mathcal{N}(v) \cup v\}} (\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}})_{v,u} x_u \Big) \forall v \in V \tag{2.20}$$

Other popular convolutional networks for graphs are:

* Graph Attention Networks (GANs) [72], that adopt a masked self-attention mechanism to learn the relative weights between two connected nodes, weighting differently the neighbors of a node.
* GraphSage [22], that adopts sampling to obtain a fixed number of neighbors for each node, thus particularly suited to be used with large graphs;
* Graph Isomorphism Networks (GINs) [76], that were proposed to implement Weisfeiler-Lehman (WL) graph isomorphism test.

2.4 Algorithmic fairness

Algorithmic fairness is one of the main topics in **AI ethics**; the definition of AI ethics may vary because there are many and different ethical principles involved, but there are five common themes across them [18]:

* *Beneficence* underlines the central importance of promoting the well-being of people and the planet with AI;
* *Non-maleficence* refers to the prevention of infringements on personal privacy and to security issues, in the sense that an AI should not harm any living being;
* *Autonomy*, the development of AI must not impact the freedom of human beings to set their own standards and norms;

* *Justice* says that AI and ML models should promote justice, trying to eliminate any kind of discrimination, granting global and equal access to its benefits;

* *Explicability* indicates the availability of answers to the questions "how does (the ML/AI model) work?" and "who is responsible for the way it works?", that refer respectively to the principles of *intelligibility* and *accountability*.

With this categorization, fairness falls under the theme of *justice* as a principle related to evaluating whether a decision is morally right; generally speaking, Algorithmic Fairness deals with the problem of developing AI-based systems able to treat subgroups (i.e. minorities) in the population equally.

These subgroups are typically determined by means of sensitive attributes, which should not be taken into account for decision purposes; in other words, fairness is the absence of any prejudice or favoritism toward an individual or group based on their inherent or acquired characteristics. Examples of these attributes are gender, age, ethnicity, and sexual or political orientation.

As an example, let's take the problem of gender-based discrimination in ads studied in [45]; the study involved an advertising campaign about STEM (science, technology, engineering, and math) jobs, implementing an algorithm that has been designed to be gender-neutral. Empirical results showed, however, that the ad was shown to more than 20% more men than women, despite not targeting a specific gender; since a recommendation algorithm improves itself by taking into account interactions with recommended items, AI-based systems may exacerbate this disparity by recursively learning that men tend to be more interested in STEM, consequently pushing towards an increased disparity.
It is then very important to understand how to enhance ML models, with fairness requirements. In fact, when these models take a *decision* in a human-oriented environment (e.g. decide whether to hire, to grant a loan, or to release on bail - as said earlier), it is ethically as well as legally discriminating to ground the choice on one or more sensitive attributes.
Recently, much attention has been paid to the societal impact of AI, especially concerns regarding its fairness. A growing body of research has identified unfair AI systems and proposed methods to debias them, yet many challenges remain.
Even the European Commission has recently published a proposal for what is going to be the first attempt ever to insert AI systems and their use in a coherent legal framework [58]: the proposal explicitly refers to the risk of bias discrimination of AI systems.

One can think that to determine if a model is fair or not, it is sufficient to define a measure based on a popular definition of fairness, apply it to the results of the model and check if it is satisfied. However, the process is not straightforward, especially if it's done without a clear explanation of why or how a particular approach was taken. Selecting a fairness definition/approach means making trade-offs and these trade-offs need to be documented in order to understand what an AI system is designed to do and why. Moreover, from a technical perspective, adding more fairness constraints places restrictions on an algorithm, resulting in lower accuracy. Also, the opaqueness of (especially deep) machine learning models can make it challenging to actually measure fairness.

In the last few years, an incredible number of definitions have been proposed, formalizing different perspectives from which to assess and monitor fairness in decision-making processes. The proliferation of fairness definitions is not per se an issue: it

reflects the evidence that fairness is a multi-faceted concept, that can involve different meanings and nuances, depending on the specific situation considered.
As is often the case with moral and ethical issues, conflicts are present between different and typically equally reasonable positions, because underlying assumptions may be subjective and derive from different philosophical points of view.

Before going into details and specifying the differences in fairness definitions and how to measure them, we can identify which are the main sources of bias in a model: most of the time, in data-driven models, bias is directly contained in data and their collection, and can lead an algorithm to even amplify and perpetuate it (as seen in the advertising example); there are also cases in which algorithms themselves are biased due to particular design choices, such as the use of certain optimization functions, regularizations, choices in applying particular statistical models on the data as a whole or considering subgroups, and unfair exploiting of user interactions.

Bias in data can be divided into two groups:

* **Statistical bias** occurs when training data are not representative of the true population; in other words, different groups may have different distributions of values of their features, that can in turn be related in different ways to the true label to which they are associated to. This can be caused by a *selection bias*, i.e., examples selected from the population comes from a non-random selection; or by *systematic measurement errors*, i.e., the past measurement will differ from the true measurement in the same direction or data can systematically miss for some part of the population.
  As an example, consider the task of predicting college performance based on high school data. Suppose there is a majority population and a minority population. The majority population takes the exam multiple times, reporting only the highest score. The minority population does not. If we train a classifier without considering the difference, to minimize overall error, if it can not simultaneously fit both populations optimally, it will fit the majority population, because it is more important to overall error [11].

  However, even in samples perfectly representative of the population it may be that some protected group happens to be a minority group: this is not the result of errors or incorrect collection, but it comes from a *true feature* of the population and it is actually one of the most popular sources of unfairness against minorities;

* **Historical bias** refers to past discrimination and inequalities. It occurs when data truly represent the population (i.e., take into account minorities and there are no collection errors), but reflects past biased decisions. In most cases, this is due to a systematic favor/disfavor towards groups of people at the moment of creating the target variable from which the model is going to learn, and we do not refer to synthetic datasets, but to real data that, because they have been classified by a human in a manual process, codify the subjective preference/way of thinking of the judge and we cannot in general trust their objectiveness and fairness.
  An example is *gender bias*, which has a rather long history, and cannot simply be exacerbated by involving new and updated data: think for example of income or profession disparities that perpetuate even in modern days; trying online searches

for "CEO" yield mostly images of white men, since senior positions are now mostly occupied by men, because of historical discrimination against women [46].

Fairness definitions are often categorized into two different groups: **statistical** and **individual** notions; the next sections will review both approaches and list some of the most used metrics in each context.

2.4.1 Group fairness

Group fairness criteria are the most popular and suitable for the idea of algorithmic fairness given in the previous section since they follow criteria that focus on equality of treatment among groups of people.
Almost all group fairness definitions fall into three categories [3]:

* *Independence* requires the model prediction to be *statistically* independent from the sensitive attribute;
* *Separation* allows correlation between prediction and sensitive attribute, to the extent that is justified by the target, in the sense that if it is known that a group is more likely to receive a negative outcome, then the fairness criteria is relaxed to allow a disparity between groups, that is, however, proportional to the real disparity of the ground truth. This is equivalent to requiring that the model has the same false positive rate and false negative rate across groups identified via the sensitive attribute. It has to be noted that this approach has the requisition to completely trust the target label.
* *Sufficiency* is satisfied when the sensitive attribute and target variables are clear from the context: the prediction contains all the information about the true label and the sensitive attribute is not needed.

The following examples of fairness metrics (defined for a classification problem with two classes) will help to better understand the three concepts.

Statistical parity

Statistical (or demographic) parity (SP/DP) is the most common metric used when measuring group fairness in machine learning [77],[12],[6],[7],[79] and it is the main representative of *independence*. Set $\hat{y} \in 0, 1$ the possible values for the prediction given by the model and $s \in 0, 1$ the possible values for the sensitive attribute, SP can be expressed as:

$$P(\hat{y} = 1|s = 0) = P(\hat{y} = 1|, s = 1) \tag{2.21}$$

It can be interpreted as the requirement that the same positive prediction ratio is given by the model, across the two groups [26].
A common variant is called *conditional demographic parity*, that given the existence of a rating R for the classification task, also considers it, requiring that people in different groups with the same rating R will have the same probability of getting a positive outcome. As an example, if a company is hiring and wants to use an algorithm for doing it, it may require that males and females will have the same probability of being hired, with the condition of being equally skilled; one must however be really careful because the variable that we are conditioning on (R) might itself be a source of unfair discrimination.

Equality of opportunity

Equality of opportunity [25] was defined to solve two issues of DP. The first is that it doesn't always ensure fairness, since the notion permits to accept qualified (i.e., with high ratings) people in one group while also accepting unqualified individuals in the other, as long as the imposed threshold is reached; the second issue is that DP cannot be applied in the learning process in cases in which the sensitive attribute is actually correlated with the ground truth since it will result in a big performance drop in term of the accuracy of the prediction.

Equality of opportunity is a representative of separation: it allows the prediction to be dependent on the sensitive attribute, but only through the target variable; in other words, it prohibits to use the sensitive attribute as a proxy for the ground truth, while encouraging the use of features to allow to predict it. Formally, it is defined as:

$$P(\hat{y} = 1|s = 0, y = 1) = P(\hat{y} = 0|, s = 1, y = 1) \tag{2.22}$$

That is, instead of considering positive rates, it requires the ratio of *true* positive rate to be equal across the two groups, punishing the model to perform well only on the majority.

Predictive Parity

Predictive parity [73] (also known as outcome test) is a representative of the last category, sufficiency. It is satisfied if for both groups the probability that a subject with a positive predictive value truly belongs to the positive class is the same. In other words, the probability of a user with a good predicted score to actually have a good score should be the same, so the *precision* should be the same across the two groups. It is formally defined as:

$$P(y = 1|s = 0, \hat{y} = 1) = P(y = 0|s = 1, \hat{y} = 1) \tag{2.23}$$

It should be noted that there are cases in which it is not possible to simultaneously satisfy different fairness metrics, despite pursuing the objective of removing disparities among the same two groups; let's formulate a toy example to make it more clear.

Consider the problem of credit lending decisions, where we want to build a predictor able to correctly classify individuals that will pay back their loan, under the constraint of not using gender information in the decision process; in other words, we want a model with high accuracy that does not discriminate on the basis of gender.

Group	**Male (s = 1)**	**Female (s = 0)**
Ground Truth (y)	1 0 1 0 1 1	0 0 1 1
Prediction ($\hat{y}$)	1 1 0 0 0 1	0 0 1 1

Table 2.1: Credit loan toy example

In Table 2.1 are reported ground truth and prediction of a hypothetic model for 10 individuals, divided into two groups by the sensitive feature *gender*; a target label of 1 indicates that an individual will repay the loan, while 0 indicates the opposite. The predictor has an accuracy of $\frac{7}{10} = 70\%$.

The probability of a man to be classified positively (PR) is $P(\hat{y} = 1|s = \text{male}) = \frac{3}{6} = 50\%$; the same probability calculated for a female is $P(\hat{y} = 1|s = \text{female}) = \frac{2}{4} = 50\%$. As both groups have the same positive prediction ratio, *DP is satisfied.*

If we consider the true positive rate (TPR), we obtain $P(\hat{y} = 1|s = \text{male}, y = 1) = 2/4 = 25\%$ and $P(\hat{y} = 1|s = \text{female}, y = 1) = 2/2 = 100\%$. As the TPR for for females is greater than TPR for males, *EO is* ***not*** *satisfied.*

Finally, the precision of the two groups is $P(y = 1|s = \text{male}, \hat{y} = 1) = 2/3 =\sim 33\%$ and $P(y = 1|s = \text{female}, \hat{y} = 1) = 2/2 = 100\%$ respectively for males and females, so also *PP is* ***not*** *satisfied.*

Table 2.2 reports an example where EO is satisfied while PP and SP are not.

Group	**Male (s = 1)**	**Female (s = 0)**
Ground Truth (y)	1 0 1 0 1 1	1 1 1 1
Prediction ($\hat{y}$)	1 1 1 1 1 0	1 0 1 1
PR	$\sim$ 83%	75%
TPR	75%	75%
Precision	60%	100%

Table 2.2: Credit loan toy example 2

2.4.2 Individual fairness

Individual fairness (IF) is embodied in the following principle: similar individuals should be given similar decisions. This principle deals with the comparison of single individuals rather than focusing on groups of people sharing some characteristics. On the other hand, group fairness starts from the idea that there are groups of people potentially suffering biases and unfair decisions, and thus tries to reach equality of treatment for groups instead of individuals. The first attempt at defining an individual fairness metric was given by Dwork et al. [16], roughly formalized as the constraint that *"similar individuals should be treated similarly"* where similarity is defined with respect to a task-specific metric that must be determined on a case-by-case basis. More formally, he introduces the concept as a Lipschitz condition on the input and model space:

$$dist_Y(\hat{y}_i, \hat{y}_j) < L \times dist_X(x_i, x_j) \tag{2.24}$$

Where $dist_Y$ and $dist_X$ denote a predefined and suitable distance metric in the target and features spaces respectively. The objective is then to learn a function that maps two examples in the input space that are close into two targets that are close; in other words, similar examples have similar outcomes. An example of individual fairness is **fairness through unawareness**.
A classifier satisfies fairness through unawarenes (FTU) if no sensitive attribute is explicitly employed in the decision process: if two people have similar non-sensitive features, they should be classified the same way. Despite being simple, it is not straight-forward to assess. A possible metric is **consistency**:

$$consistency = 1 - \frac{1}{n}(\sum_{i=1}^{n} 1|\hat{y_i} - \frac{1}{k} \sum_{x_j \in kNN(x_i)} \hat{y_i}) \tag{2.25}$$

Where kNN(x_i) indicates the k nearest neighbors of an example x_i, calculated via the selected distance metric. For each observation $(x_i, \hat{y_i})$, it measures how much the predicted decision $\hat{y_i}$ is close to the outcome for the neighborhood of the example x_i. Sometimes this definition can however increase unfairness in the model: if for example, a man has only other men in his neighborhood, then requiring the prediction to be the same for all of them is equivalent to saying that all men should be classified with the same value; in this case, even with consistency equal to 1, the model is not stopped from using the sensitive attribute.

Another problem with FTU is that it does not consider possible correlations between non-sensitive features and sensitive ones. For example in a particular dataset, it can be that men and women have different values for some non-sensitive features because of natural and legitimate differences between the two genders; in this case, the difference in those features can be used as a proxy to derive the sensitive attribute even if it was removed.

A common method used in literature to deal with correlations is to "clean" the dataset to hide sensitive and related attributes in other features and takes the name of **suppression**; this approach has however the main problem of the potential loss of legitimate (i.e., useful for the accuracy on the downstream task) information that may be intrinsic in features related to the sensitive one. An example of an approach that tries to solve this last issue is projecting the features into an orthogonal feature space; it may be however difficult to consider all possible feature interactions.

Although the semantics of these kinds of definitions can be more intuitively "right" than statistical approaches to fairness, since "similar treatment" can be a reasonable definition of fairness, there are various difficulties encountered when trying to implement such appealing definitions. The main issue is that they require making a significant assumption on the context in which the model has to be applied and on what distance metric should be used to be really fair w.r.t the task, a process that often requires help from a domain expert; moreover, there are cases in which two examples that are similar are also similarly discriminated: an approach based only on IF can potentially amplify the discrimination against those examples. This argument is studied in detail in [17].

Therefore, using only IF measures to assess the fairness of a model can not be sufficient; there are however studies that try to satisfy both individual and group fairness simultaneously, by taking the best of both worlds. Kearns [37], for example, proposes an approach that informally states that *"on average, individuals in two groups should be treated similarly if on average the individuals in the two groups are similar"*. Another example can be the integration of *consistency* in the implementing process of a group-based fair model: if we have a trusted distance metric, appropriately calculated without considering sensitive attributes, then we can use consistency to understand if and how the model tends to move away examples that instead should be treated similarly because sharing many values for the non-sensitive feature; this is especially useful in black-box models that are hardly understandable.

In any case, when trying to integrate IF measures, GF metrics should have a stronger impact on the model because of the already cited problems that individual fairness can introduce: current definitions of individual fairness metrics seems to be more suitable to understand the behavior of a classification model from a different point of view, rather than to directly assess the discrimination of its results. Table 2.3 summarizes the fairness definitions defined previously.

It must also be noted that often individual and group fairness definitions appear to be in conflict; however, as demonstrated in [5], they do not always reflect different principles, since they can instead answer the same question and follow the same underlying assumption on what fairness should grant.
Moreover, in [16] has been demonstrated that both statistical parity and individual fairness can be satisfied with the correct choice of distance measure.

	Notion	Satisfied if
Group fairness	Statistical parity	Same positive prediction ratio
	Equality of opportunity	Same *true* positive prediction ratio
	Predictive parity	Equal precision
Individual fairness	Fairness through unawareness	No sensitive attribute used in the decisiond

Table 2.3: Different fairness notions

2.5 Applying fairness

Once a fairness definition is chosen, there are three main strategies to impose it in a machine learning model [56],[51]: *pre-*, post- and in- processing.
Pre-processing refers to techniques to transform the input data before feeding them to the machine learning model, in a way that can remove bias directly on data; this can obviously be done only when the algorithm is allowed to modify the training data.
In-processing refers to techniques that operate on state-of-the-art models that are known to have high accuracy, modifying their training procedure in order to remove possible discrimination, typically by modifying the optimization function or by imposing some constraints.
Lastly, *post-processing* refers to techniques applied after training a model that tries to increase fairness on the given solution by reassigning labels based on a defined function; in other words, it modifies posteriors to satisfy fairness constraints.

In the next chapter, we will review some state-of-the-art models and see some examples of the different approaches.

Chapter 3

Related works

Recently, fairness transformed from a niche topic with a low number of papers produced every year, to a major subfield of machine learning.
Moreover, modern Deep Learning techniques permit nowadays to create models that are much more effective when operating with complex data such as graphs, by learning directly from data; the problem is that this high performance on data also increases the risk of inheriting human historical biases hidden in past data. Since the performances of those models on graph data have increased pretty recently compared to models on standard data, the literature is not as rich as it is for other settings, but there are some works that should be cited.

Although not specifically talking about fairness, an early work on the topic comes from a 2016 paper [74], in which the authors analyze whether moving to network-based data from individual's data could influence the result in credit scoring; the work deals with the problem of introducing social contacts information in evaluating individual creditworthiness (that is a topic many research groups are working on [49],[42],[34],[35]), and it comes natural to make a parallel with information aggregation given by GNNs. The result of their analysis showed that introducing social network information can potentially increase discrimination against already financially disadvantaged groups.

A more recent work proposing a method to implement group fairness on learning with graphs is fairwalk [61]; the authors modify the node2vec [20] algorithm for generating walk traces in a graph: as explained in 2.3.1, this procedure starts from a node, then jumps to another neighbor node multiple times and returns a list of traces that can be used to generate a new representation. Instead of randomly jumping to neighbors, *fairwalk* divides the nodes into groups based on their sensitive attribute and assign the same probability to be chosen for the jump to every group; by doing so, each subgroup has the same probability of being explored from the algorithm, giving more visibility to minorities.

Another often cited early work is *fairgnn* [12]. This model does not include sensitive attributes in the learning process, adopting adversarial learning where the adversary aims to estimate the sensitive attribute while another module aims to learn fair node representations. Similarly, Bose et al.[6] propose a model that learns a set of adversarial [48] filters that try to remove information about the sensitive attribute. The authors state that their approach is also compositional, in the sense that it can

generate different embeddings for different selected sensitive attributes or combinations of sensitive attributes.

This thesis modifies and extends a state-of-the-art method called *Nifty*[1], inspired by two other works that demonstrated good results in term of fairness, namely *FairDrop*[70] and *FairRF*[79]; next section will describe those three models.

3.1 Nifty

The main objective of Nifty is to learn a *fair* and *stable* representation.
We already talked about fairness in section 2.4, so we should briefly define the concept of stability, even though it is a problem not studied in this thesis. An ML algorithm is said to be stable if it produces consistent predictions with respect to small perturbations of training samples; it is an important characteristics because unstable predictions can harm the reproducibility of the results, other than reducing the user's trust in the model.
The main idea starts from a connection between counterfactual fairness(CF) [43] and stability. To satisfy CF, a model must assign the same classification probability to two examples that share all features except the sensitive one; in other words, the model should be *stable* w.r.t. modifications to the sensitive attribute. Leveraging this connection, the authors propose to minimize the similarity between the original nodes' representations and their *augmented* (i.e. perturbed) counterpart, with the objective of making the model stable to the noise introduced by modifying the input, thus able to distinguish relevant information.
Three different augmentations (and counterparts) are produced:

1. Perturbing node's attribute (except the sensitive one for which we want to guarantee fairness); the vector of augmented attributes is defined as $\bar{x_v} = x_v + r \cdot d$, where $d \in \mathbb{R}^M$ is sampled from a normal distribution and $r \in \{0,1\}^M$ is a masking vector draw from a Bernoulli distribution;

2. Perturbing node's sensitive attribute, creating a *counterfactual* by flipping the value of s_u from 0 to 1 and vice-versa;

3. Perturbing graph structure, by constructing a new adjacency matrix $\hat{A} = A \cdot R_e$, where $R_e \in 0, 1^{N \times N}$ is a random binary mask drawn from a Bernoulli distribution $\mathbb{B}(1 - p_e)$, where p_e is the probability of dropping an edge.

Nifty infuses fairness and stability in the objective function, as well as in the architecture of underlying GNN:

* Objective function: it introduces a triplet-based objective that maximizes the agreement between the original graph and its counterfactual and noisy views, building a siamese network using node, sensitive attribute, and edge level perturbations. Then it trains a GNN encoder using the Siamese framework; the encoder generates representations z_v of the augmented graph at every iteration. By generating augmented graphs, NIFTY can induce appropriate bias into the underlying GNN to learn embeddings that are invariant to the combination of counterfactual nodes as well as to random perturbations in the graph structure.

* Architecture: NIFTY modifies the GNN's routing of neural messages, through Lipschitz normalization of the weight matrix at each layer, by dividing it by its

spectral norm; this permits to limit the difference between original and augmented embeddings;

The siamese GNN encoder generates the representations of each graph node $\tilde{z}_v$ and its augmented version $\bar{z}_v$. Then a predictor $t : \mathbb{R}^d \to \mathbb{R}^d$ is used to transform and match corresponding representations.
The training minimizes the following loss function:

$$L = \min_{\Theta} \mathbb{E}_v[(1-\lambda)L^c] + \lambda L^s, \tag{3.1}$$

where L^c is the Binary Cross Entropy (BCE) loss, Θ are the trainable parameters, and L^s is a triplet-based objective function that optimizes the similarity between augmentations of the same node, by reducing their cosine distance. It is defined as

$$L^s = \mathbb{E}_v[\frac{1}{2}(D(t(z_v), sg(\bar{z}_v)) + D(t(\bar{z}_v), sg(z_v))], \tag{3.2}$$

where D is the cosine distance.

3.2 Fairdrop

Fairdrop [70] is a biased edge dropout algorithm to counter-act homophily (the principle that similar users interact at higher rate than dissimilar ones) of the sensitive attribute to improve fairness in graph embeddings.
The intuition comes from the fact that the homophily of sensitive attributes affects the prediction in the information aggregation process: learning a representation for nodes that are close in the graph structure will involve a neighborhood composed of many shared nodes. This means that if nodes in the same sensitive group are close together, their embedding will be similar and will influence the model into classifying them in a similar way: an algorithm that does not take this behavior into consideration will probably emphasize differences between groups, learning a representation that increases their distance in the output space. For example, in [21] authors show that users in a social network belonging to major political groups are exposed to more information more quickly w.r.t. minorities.
Technically, for every epoch of training, authors propose to:

1. Compute an $n \times n$ mask M encoding the edge homophily w.r.t. a sensitive attribute, i.e. $m_{i,j} = 1$ if $s_i \neq s_j$, 0 otherwise,

2. define $\tilde{M}$, a random perturbation of M, as

$$\tilde{m}_{i,j} = \begin{cases} m_{i,j} & \text{with probability: } \frac{1}{2} + \delta \\ 1 - m_{i,j} & \text{with probability: } \frac{1}{2} - \delta \end{cases} \tag{3.3}$$

3. drop edges from the original adjacency matrix according to the computed perturbation: $A_{fair} = A \circ \tilde{M}$.

With this pipeline, authors train the model over different and fairer random copies of the adjacency matrix, increasing the randomness and diversity of the input. The *delta* parameter regulates the level of fairness enforced: when $\delta = 0$, it drops random

edges with a probability $p = \frac{1}{2}$, while when $\delta = \frac{1}{2}$, it drops **all** the homophilous edges, keeping only the heterophilous ones. Authors show that removing homophilous edges has a positive effect on different fairness metrics, because disconnecting or moving away nodes that share the same sensitive attribute reduces the importance of that attribute in the computation of the resulting node embeddings.

3.3 FairRF

The third and last work that inspired this thesis is *FairRF* [79]. This work isn't actually related to graphs but proposes a general method that can also be applied with the complex data that we work with, also because it is designed to be flexible to use different classifiers as the backbone.

The authors consider a problem where the sensitive attribute is not available, working instead on features that are known to be correlated with the sensitive one. Formally, they define $\mathbb{F}_s$ the set of non-sensitive features that are correlated with a sensitive s and $\forall f_j \in \mathbb{F}_s$ a correlation regularizer is applied, with the purpose of making the model fair towards s. The regularization term is the following:

$$\min_{\Theta} \mathbb{R}_{related} = \sum_{j=1}^{K} \lambda_j \times \mathbb{R}(x^j, \hat{y}) \tag{3.4}$$

Where λ_j is the weight for regularizing correlation coefficient between x^j and $\hat{y}$. $\mathbb{R}(x^j, \hat{y})$ is defined as:

$$\mathbb{R}(x^j, \hat{y}) = |\sum_{i=1}^{n} (X_{ij} - \mu_{X^j})(\hat{y}_i - \mu_{\hat{y}})| \tag{3.5}$$

where μ_{X^j} is the mean of x^j.
The reason to use such weights is to reduce the correlation between sensitive attribute and predicted value by increasing the importance of the regularization term for attributes with high correlation with the sensitive one.

A limitation of this approach is the requirement of some pre-defined values for λ; since in real-world applications having accurate values for those weights can be difficult, the training procedure also includes their update.

Formally, the entire optimization function is defined as:

$$\begin{aligned} \min_{\Theta,\lambda} \quad & \mathbb{L}_{cls} + \eta \sum_{j=1}^{K} \lambda_j \mathbb{R}(x^j, \hat{y}) + \beta ||\eta||_2^2 \\ \text{s.t.} \quad & \lambda_j \geq 0, \forall f_j \in \mathbb{F}_s; \sum_{j=1}^{K} \lambda_j = 1 \end{aligned} \tag{3.6}$$

Where η sets the weights of the regularization term and $\mathbb{L}_{cls}$ is the classification loss, β is used to control the contribution of $||\eta||_2^2$.
Updating θ (parameters of the classifier) and λ is done iteratively: at each step, one of the two is updated while the other remains fixed.

In their results, the authors show how their method is able to reduce unfairness while sacrificing very little classification accuracy.

3.4 Other works

This section briefly describes other recent works to offer a wider view of state-of-the-art methods.

EDITS [14] proposes a framework composed of three modules:

1. Attribute Debiasing. This module learns a debiasing function $g_\Theta$with learnable parameter $\Theta \in \mathbb{R}_\mathbb{M}$. The debiased version of X is obtained as output where $\bar{X} = g_\Theta(X)$.

2. Structural Debiasing. This module outputs $\hat{A}$ as the debiased adjacency matrix A. Specifically, $\hat{A}$ is initialized with A at the beginning of the optimization process. Its entries are then optimized via gradient descent with clipping and binarization.

3. Wasserstein Distance Approximator. This module learns a function f for each attribute dimension. Here f is utilized to estimate the Wasserstein distance between the distributions of different groups for any attribute dimension.

REDRESS [15] is one of the few works that use an *individual fairness* notion. Given the oracle pairwise similarity matrix S_G of the examples in the input graph G, and the similarity matrix S_Y among instances in the outcome space (defined upon a proper similarity metric), to relieve the individual bias toward fair decision-making, they say that the predictions are individually fair if for each example x, the two ranking lists that encode the relative order of other examples are consistent.

Debayes [7] is an adaptation of Conditional Network Embedding (CNE) [36], a Bayesian approach based on integrating prior knowledge through prior distribution for the network.

[47] creates a fairer adjacency matrix following the concept of dyadic fairness for the task of link prediction.

MCCNifty [78] is another extension of nifty; this work uses the same perturbation process of Nifty, but instead of doing it in the entire graph, a multi-view extractor based on *Pagerank* [57] extracts informative regional structures (subgraphs) to which the perturbations are done.
Authors justify their work by the fact that considering the entire graph structure is computationally expensive, can be affected by global bias, and does not consider the problem of *overconfidence*, i.e., can assign wrong predictions to very noisy examples or out of the distribution.

Influence maximization is also a quite popular approach used, see [71], [39].

Chapter 4

Proposed methods

As introduced in chapter 3, this thesis proposes to extend the state-of-the-art method [1], inspired by related literature ([70], [79]) shown to be empirically effective. We've been motivated by the fact that, although providing good results in the fairness metrics, NIFTY followed an approach purely based on randomization; instead, we show that we can formulate some heuristics that can further increase the effectiveness of the method in reducing bias and unfairness of its results.

More specifically, we focused on the graph perturbations applied in the encoding process, producing two different extensions of NIFTY:

* Biased edge dropout in graph structure perturbation;

* Biased non-sensitive attribute perturbation;

In their work, NIFTY's authors used different GNN versions as the backbone for their model; we instead focused only on the simple yet effective GCN formulation, mainly for reasons of time, with the hope to extend our work to other formulations in the future. More specifically, the GCN used in this work is composed of a convolutional layer with 16 hidden units, followed by a linear fully connected layer; we choose this architecture to have comparable results because it is the same used by NIFTY as GCN backbone. This architecture can be seen as a starting point, able to give good results while not being too complex; a more accurate tuning can be made, however, it is reasonable to think that using many more hidden units and/or other adding more hidden layers could impact too much the complexity of the model since the overall method is composed of different modules, each of which contributes in the overall complexity. Moreover, this work mainly focuses on fairness enforcing method, so we do not focus on directly increasing the accuracy.

Since both extensions performed well for the analyzed datasets in reducing unfairness, a third natural extension is the combination of the two; the next sections will describe the proposed approaches and the solutions implemented for both models and their combination.

4.1 Biased edge dropout

Since NIFTY (see section 3.1) randomly drops edges that connect same sensitive attribute's node proportionally to their distribution in the graph, the resulting perturbed adjacency matrix will have approximately the same homophily level as the original one; it has been shown that [70] using a perturbation that is biased toward removing homophilous edges can be beneficial for the fairness of the learned model (see section 3.2).

With the objective of involving different and fairer augmented versions of the adjacency matrix in the information aggregation process, we implemented a softer version of FAIRDROP injected in the structure perturbation process; in fact, FAIRDROP operates on the whole graph without the need to specify a drop rate, only using the hyperparameter δ to indicate how much percentage of homophilous links to remove from the graph (as defined in equation 3.3): this means that the actual number of dropped edges will depend on how many of them are connecting same attributes' nodes.

In this work, we instead follow NIFTY's pipeline, where the drop rate must be specified; more specifically, we apply the default dropping made by the method and then adjust the ratio between fair and unfair eliminated edges using the hyperparameter δ; starting from the result given by the default drop, we can obtain a solution as close as possible to the original one, without introducing too much extra randomness in the augmentation process.

More formally, given the augmented adjacency matrix $\tilde{A}$ constructed by NIFTY for a certain drop rate, we define:

$$E^{\tilde{A}} = \{(i,j)|(i,j) \in E, \tilde{a}_{i,j} = 1\} \tag{4.1}$$

as the set of edges not dropped and:

$$E^{\neg\tilde{A}} = \{(i,j)|(i,j) \in E, \tilde{a}_{i,j} = 0\} \tag{4.2}$$

as the set of edges dropped.

We consider $E^{\neg\tilde{A}}$ and the hyper-parameter δ and check whether in the set there is at least a ratio of $0.5 + \delta$ edges connecting nodes with the same value of the sensitive attribute (homophilous). If not, we randomly replace some *heterophilous* edges (i.e., edges connecting nodes with different value for s) from $E^{\neg\tilde{A}}$ and insert them in $E^{\tilde{A}}$, replacing them with *homophilous* edges from $E^{\tilde{A}}$; in other words we reconnect some heterophilous links, dropping the same number of (random) homophilous ones, taken from $E_{\tilde{A}} \setminus E_{\neg\tilde{A}}$. The number of replaced nodes is given by the difference between the actual ratio of homophilous edges dropped and the given $0.5 + \delta$. Algorithm 1 reports the pseudo-code used for this process.

Clearly, when in the graph structure the percentage of homophilous edges is already bigger than $0.5 + \delta$ % our extension doesn't activate and reduces itself exactly to NIFTY. On the contrary, the higher the value of δ, the more homophilous links are removed from the adjacency matrix. In this way, the augmented adjacency matrix will contain fewer homophilous links, while maintaining the majority of the edges in the original perturbation from NIFTY: the main difference is given by the selection of

edges added and removed since it does not re-select which edges to remove.

Algorithm 1 Biased edge perturbation

Input Adjacency matrix $\mathbf{A}$, sensitive attributes vector $\mathbf{s}$, δ, drop rate p

Output Perturbed adjacency matrix $\tilde{A}$

$M \leftarrow$ *boolean mask as defined in 3.1*

$R \leftarrow A[\neg M]$ *// apply negation of the mask to obtain edges removed by standard NIFTY*

$HO, HE \leftarrow$ *homofilous, heterophilous edges* $\in R$

if $\frac{|HO|}{|HO+HE|} < 0.5 + \delta$ **then**

  $sizeA = 0.5 * |a_{i,j} \in A | a_{i,j} = 1|$

  $diff \leftarrow (\delta - p) * sizeA$ *// number of edges to replace*

  $keptEdges \leftarrow \{a_{i,j} \in A | m_{i,j} \in M, m_{i,j} = True\}$

  $droppedEdges \leftarrow \{a_{i,j} \in A | m_{i,j} \in M, m_{i,j} = False\}$

  $removableEdges \leftarrow keptEdges \cap HO$

  $addableEdges \leftarrow droppedEdges \cap HE$

  $sizeAdd = |addableEdges|$

  $sizeRem = |removableEdges|$

  if $sizeRem <$ *diff or* $sizeAdd <$ *diff* **then**

    $diff = min(sizeRem, sizeAdd)$

  $ind_{rem} \leftarrow$ first diff elements of *random permutation of* $removableEdges$

  $ind_{add} \leftarrow$ first diff elements of *random permutation of* $addableEdges$

  $M[ind_{rem}[:]] \leftarrow True$

  $M[ind_{add}[:]] \leftarrow False$

return $A[M]$

4.2 Biased attribute perturbation

Another procedure in which NIFTY introduces some randomness to enforce stability and fairness in the model is the *attribute perturbation* (see section 3.1), used to generate one of the augmented representations. Even in this case, we start from the idea that, while random modifications in the values of the attributes could enforce stability and consequently fairness because of the connection between the two introduced in section 3.1, a premeditated biased perturbation could drive the model to a fairer learning. In particular, we would like our model to perturb with greater probability attributes that could impact the use of sensitive attributes in the learning process, i.e. features that are correlated with the sensitive one. The third perturbation used in NIFTY (see section 3.1) does something similar, by flipping the sensitive attribute to generate a sort of counterfactual augmentation; however, as discussed in subsection 2.4.2, one of the reasons why removing the sensitive attribute may not be sufficient is exactly the fact of not considering the possible relationships between that attribute and the non-sensitive ones: flipping the sensitive attribute is not different from removing it, because it tries to force the model into learning that the feature is not relevant in the classification task, i.e. it should not be used in the prediction.

For this reason, we analyzed the literature looking for a method or idea that could help in our problem; we found a possible solution in FairRF [79], a method that tries to mitigate unfairness by exploring the correlations of the sensitive feature with the non-sensitive ones. This method however pre-defines some related features by exploiting previous knowledge about the topics of the datasets, supported by relative literature and empirical results. We do not have this privilege, so we decided to exploit the correlation between features found by a previous relationship analysis. More specifically, we first calculate a correlation coefficient between the nodes' sensitive attribute and the other attributes, then fix a threshold on its absolute value, in order to select only the most relevant ones, i.e. the ones that exhibit a greater correlation.

We follow the correlation weights update procedure implemented in the original paper, but we do not include a correlation loss in our optimization function, because differently from FairRF, we already have a process that implicitly influences the loss computation, i.e. the attribute perturbation; in fact, we use weights in the attribute modification process to force the model to perturb related features with a probability higher than the probability of perturbing other attributes; this probability will be proportional to the value of the weights, i.e., a related attribute with high weight will have more chance of being perturbed.

Formally, we define $\mathbb{F}_s$ as the set of related attributes with a correlation coefficient greater than the threshold τ, $\Lambda = \lambda_j \forall f_j \in \mathbb{F}_s$ as the set of weights for the related features and $\Omega = \omega_j, \forall f_j \in \mathbb{F}_s$ as the set of absolute values of correlation coefficients for the related features.

We start by initializing Λ with the values in Ω, so that for each feature $f_j \in \mathbb{F}_s$, $\lambda_j = \omega_j$; we than inject those weights in the attributes perturbation function to modify the perturbing probability assigned to related features.
In the default augmentation procedure defined in 3.1, the masking vector $r \in \{0,1\}^M$ was drawn from a Bernoulli distribution, i.e., $r \sim \mathbb{B}(p_n)$, with p_n the probability of independently perturbing an attribute; in our extension, this procedure remains the

same for non-related features, while the perturbing probability for related attributes becomes $p_n + \lambda_j, \forall f_j \in \mathbb{F}_s$. The new value of a perturbed attribute is calculated the same way defined in 3.1, i.e., $\bar{x_v} = x_v + r \cdot d$, where $d \in \mathbb{R}^M$ is sampled from a normal distribution and $r \in \{0, 1\}^M$ is the masking vector.

Chapter 5

Experimental evaluation and discussion

This chapter reports the experiments relative to the methods described in the previous chapter, along with a description of used datasets and metrics. Results show that our proposals can improve different fairness metrics compared to the original NIFTY formulation while maintaining the same computational complexity and the same level of predictive performance.

5.1 Datasets

We use the three datasets used by NIFTY, along with the well-known *Pokec* [12] dataset; for the first three, in addition to what the authors of NIFTY reported, we also include the level of homophily of each graph, that is the rapport between the number of edges connecting nodes with the same sensitive attribute and the total number of edges, since it's a fundamental characteristic to understand the behavior of our *biased edge perturbation* 4.1 extension.

* **German credit graph** (*German*) has 1000 nodes representing clients in a German bank, connected based on the similarity of their credit accounts; clients are to be classified as good or bad credit risks and the sensitive attribute is their gender. It has 22242 out of which 17998 are homophilous of them connect nodes with the same value for the sensitive attribute, so it has an homophily level of 80.92%.
* **Recidivism graph** (*Bail*) has 18876 nodes representing defendants who got released on bail, collected from several state courts in the US between 1990-2009; nodes are connected based on the similarity of their criminal records and demographics; defendants are to be classified as releasable or not releasable and the sensitive attribute is their ethnicity (black or white). It has 321308 edges and an homophily level of 53.61%.
* **Credit defaulter graph** has 30000 nodes representing individuals, connected based on the similarity of their spending and payment patterns; the task is to predict whether an individual will default on the credit card payment or not and

the sensitive attribute is their age. It has 1436858 edges and a homophily level of 95.99%.

* Finally, we consider a dataset where, differently from the others, edges are not generated based on some heuristics, but are taken from real world data: **Pokec** [12] is a dataset representing 66569 users in a social network, connected by friendship relationships; the task is to predict the working field, and the sensitive attribute is their region. It has 550331 edges, with an homophily level of 95.58%.

In *German*, *Bail* and *Credit*, the similarity measure used to calculate the similarity between pairs of nodes is the *Minkowski distance*, that for two variables a and b is defined as:

$$minkowski\,(a,b) = \left(\sum_{i=1}^{n} |a_i - b_i|^p\right)^{\frac{1}{p}} \tag{5.1}$$

Where p can take any value greater than 1; in their implementations, the authors use $p = 2$, which corresponds to use the *Euclidean distance*. The complete equation used is:

$$\frac{1}{(1 + minkowski(a,b))} \tag{5.2}$$

For each dataset, the authors set a different threshold to decide which nodes to connect, calculated as a percentage of the maximum similarity between all respective nodes; in particular, for *bail* is 60%, for *german* is 80% and for *credit* is 70%.

Table 5.1 reports main characteristics of each dataset.

Dataset	German	Bail	Credit	Pokec
Nodes	1000	18,876	30,000	66,569
Edges	22,242	321,308	1,436,858	729,129
Node attributes	27	18	13	59
Sensitive	Gender	Race	Age(≤ 25/ > 25)	Region
Labels	good credit	bail	payment default	Working field
Homophily level	80.92%	53.61%	95.99%	95.58%

Table 5.1: Datasets details

5.2 Experimental setup

Experiments have been preliminarily run on Google Colabolatory [1]; then, since the models required quite a lot of time to be trained and the service doesn't allow more than 12 consecutive hours on the standard plan, we started using a personal machine running Ubuntu 22.04, with an Intel-10700k CPU, a NVIDIA RTX 3070 GPU and 16GB of RAM.
Models have all been developed in Python 3.7, using PyTorch [2] 1.7.1+cu11 and

[1] Google Colabolatory [29]
[2] PyTorch [59]

PyTorch-Geometric [3] 1.7.0 libraries. Code developed during the thesis period will be made available in author's github page [4].

5.2.1 Baselines and configurations

To evaluate the effectiveness of our implementations, we used 4 baselines 3:

* **NIFTY**: since this is our main baseline, we kept most of the hyperparameters reported by the authors in the paper, i.e.: Adam optimizer with a learning rate of $1e^{-3}$, weight decay $1e^{-5}$ and number of epochs 1000 for all datasets except *German*, for which we increased the number of epochs to 2500 to obtain results closer to the ones reported in literature; some other hyperparameters varied for different datasets, more details will be given in the next section.
* **FAIRDROP**: we tested the performance of the base model using different values for δ (see section 3.2), but we substituted the backbone GCN with the same one used in NIFTY to make the comparison reasonable; it should also be noted that FAIRDROP has been implemented to perform the task of *fair link prediction*, not *node classification*, so an adaptation to work with our setting and datasets was necessary; optimizer setting and epochs are the same used for NIFTY.
* **Linear regression**: we implemented a simple linear regression model to check how considering only nodes' attributes would have affected the performances, mainly in terms of fairness;
* **GCN**: we also performed some experiments using the backbone GCN alone (configured as described in chapter 3), to check the effective fairness reduction performances of the other methods.

5.3 Metrics

5.3.1 Area under the curve

To evaluate the *predictive performance* for the downstream task of node classification we use *Area under the curve* (**AUC**).

Results obtained by a classificator during testing are counts of the correct and incorrect classifications of each class; this information can be displayed in a *confusion matrix*, that shows the differences between true and predicted classes for the set of labeled examples in the test set. Those values allow for the calculation of the *True positive rate (TPR)* and *False positive rate (FPR)*, defined as follows:

$$TPR = \frac{true\ positives}{true\ positives + false\ negatives}$$

$$FPR = \frac{false\ positives}{false\ positives + true\ negatives}$$

Those parameters are plotted in a graph called **ROC** (receiver operating characteristic) curve at different classification thresholds, showing the trade-off between

[3] PyTorch Geometric [60]
[4] Author github [30]

TPR (y-axis) and FPR (x-axis): lower thresholds classifies more examples as positive, increasing both false and true positives.
AUC aggregates the performance across all possible thresholds, measuring the 2-dimensional area underneath the ROC curve: it ranges from 0 to 1, telling how much the model is able to distinguish between classes. Figure 5.1 shows a graphical example of the metric.

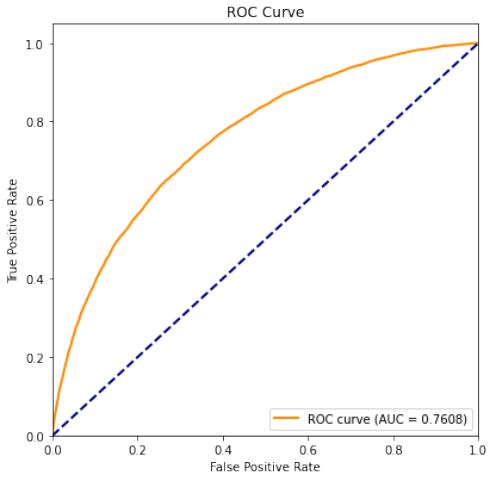


Figure 5.1: ROC and AUC example

5.3.2 Fairness metrics

To measure fairness, we use the same group metrics used in NIFTY, i.e. **Statistical Parity** and **Equality of Opportunity (EO)** (both described in section 2.4.1), where probabilities are estimated on the test set we use ΔSP, ΔEO and CF to indicate.

We also use the **Counterfactual fairness (CF)** metric, where the score is calculated as the percentage of test nodes for which the predicted label changes when *flipping* the node's sensitive attribute. In line with related literature, we report the fairness metrics as percentages from 0% (never discriminating) to 100% (always discriminating), so the lower the better. We use ΔSP, ΔEO and CF to indicate Statistical Parity, Equality of Opportunity, and Counterfactual Fairness respectively, in terms of the difference between probability for the two groups.

Results are presented in form of tables, where we report mean and standard deviation of 3 to 5 independent runs for each random seed used (0,1 and 2).

5.4 Results

5.4.1 GCN and linear regression models

This section reports the results obtained by using the GCN backbone alone and a simple linear regression model. In both we use the same optimizer used in NIFTY; since we needed to find a way to debias the model by also working on node's attributes,

we test both by including or not the sensitive attribute, to see if and how much it could affect performances in terms of fairness.

Since Counterfactual Fairness explicitly requires the sensitive attribute to generate flipped versions of the examples, the results of the models trained without it do not report any value for that metric.
Table 5.2 reports the results with the GCN for German, while Tables 5.4 and 5.3 for Bail and Credit.

Results on German (Table 5.2) clearly show that an approach that only relies on hiding the sensitive attribute isn't sufficient to reduce unfairness: it even increases the unfairness of the results that we obtained, however, being the values so high, it should not be a direct consequence of removing the feature, but more probably an effect of randomness in the learning process. Another important observation is that those results demonstrate the effectiveness of NIFTY and in particular of our extension.

Results on Bail and Credit (Tables 5.3 and 5.4) show instead a more expected behavior: removing the sensitive attribute slightly reduces accuracy, but also decreases the unfairness of the learned representation; both SP and EO are lowered, remaining however higher than NIFTY for Bail. For Credit without sensitive, instead, results are very similar to the ones obtained by our extensions: it can be a bit discouraging considering the complexity of NIFTY and our extension compared to a "simple" GCN, however, it demonstrates that even without excluding the sensitive attributes, our model is able to learn a solution that somehow tries to not use that feature.

	German			
Attribute	AUC	ΔSP	ΔEO	CF
SENSITIVE	69.32 ±0.52	40.01 ±1.89	40.86 ±3.94	14.13 ±1.5
NO SENSITIVE	69.42 ±0.27	43.99 ±0.29	46.49 ±2.45	

Table 5.2: Backbone GCN w and w/out sensitive attribute with German dataset

	Bail			
Attributes	AUC	ΔSP	ΔEO	CF
SENSITIVE	93.92 ±0.0	7.27 ±0.02	4.7 ±0.01	3.9 ±0.0
NO SENSITIVE	93.91 ±0.0	5.73 ±0.02	4.01 ±0.0	

Table 5.3: Backbone GCN w and w/out sensitive attribute with Bail datasets.

	Credit			
Attributes	AUC	ΔSP	ΔEO	CF
SENSITIVE	73.93 ±0.0	12.92 ±0.01	10.64 ±0.01	1.38 ±0.0
NO SENSITIVE	73.89 ±0.0	11.72 ±0.0	9.42 ±0.0	

Table 5.4: Backbone GCN w and w/out sensitive attribute with Credit datasets.

Table 5.5 reports the results with the linear regression model for German, while Tables 5.6 and 5.7 for Bail and Credit.

The linear regression model performs very well in terms of accuracy. It also *looks like* edges do not bring any relevant information for the accuracy of the predictions, however, the backbone GCN used is really simple and not comparable to the best performances models that can be found in literature.
For what concerns the fairness metrics, we see the same exact behavior seen with the GCN alone, with the difference that in Credit, SP and EO are way lower than all other models tested. This may be caused both by the setting of the GCN, which remained fixed and has not been accurately tuned (to remain coherent with NIFTY), and by the fact that, as said before, edges are built and do not refer to a real-world graph; moreover, Credit has 47 times more edges than nodes: this may cause the model to give a lot of relevance to the links that, despite providing additional information, may introduce some noise that more complex (or deeper) architectures could possibly recognize and remove.

In any case, it is clear that removing the sensitive attribute isn't always effective in mitigating unfairness, especially in German where we get the opposite behavior. For this reason, we implemented an extension to work on non-sensitive attributes whose results are presented in the next section.

	German			
Attributes	AUC	ΔSP	ΔEO	CF
SENSITIVE	66.95 ±4.34	5.87 ±4.72	5.65 ±3.86	2.7 ±2.49
NO SENSITIVE	69.1 ±2.56	7.63 ±4.84	5.36 ±4.0	

Table 5.5: Linear regression w and w/out sensitive attribute with German dataset

	Credit			
Attributes	AUC	ΔSP	ΔEO	CF
SENSITIVE	72.77 ±0.44	4.3 ±0.51	2.32 ±0.48	1.1 ±0.21
NO SENSITIVE	72.75 ±0.41	3.03 ±0.49	1.5 ±0.46	

Table 5.6: Linear regression w and w/out sensitive attribute with Credit datasets

	Bail			
Attributes	AUC	ΔSP	ΔEO	CF
SENSITIVE	97.32 ±0.24	8.07 ±0.36	4.89 ±0.48	0.91 ±1.5
NO SENSITIVE	97.37 ±0.16	7.26 ±0.13	2.7 ±0.5	

Table 5.7: Linear regression w and w/out sensitive attribute with Credit datasets

5.4.2 Biased attribute perturbation

This section reports results obtained with the extension described in section 4.2.
This extension proposes to modify the attribute perturbation process of NIFTY by assigning higher perturbation probability to attributes related to the sensitive one. First of all, to understand if working only on nodes' attributes could help mitigate unfairness in the solutions of a model trained on our dataset, we performed some preliminary experiments with two simpler models: the *backbone GCN* used in NIFTY and a *linear regression* model, both with and without including the sensitive attribute; the former helped us understand the actual effectiveness of NIFTY, while the latter, since a linear regression cannot use information coming from edges, highlighted how much the graph structure could help the learning model, both in terms of accuracy and fairness metrics. Results in subsection 5.4.1 show that in both cases, removing the sensitive attribute is not enough to make the model fair because, as discussed in 2.4, fairness through awareness usually fails in reducing bias in a machine learning model.

The first reason that came to our mind was that probably there were other features somehow correlated with the sensitive one, that could potentially be exploited by a model to derive the unknown sensitive attribute as proxies; to verify this thesis, we performed a brief study on the relationship between different attributes in our datasets. In practice, for each dataset, we built a correlation matrix using the *Spearman correlation coefficient*, since it can evaluate monotonic relationships between variables, instead of only on linear relationships like the *Pearson* coefficient, so can potentially capture more information. Figures 5.2, 5.3 and 5.4 show the obtained matrix in form of heat-maps (the image for *Pokec* is omitted since its nodes have too many attributes to be correctly displayed); in the figures, the row and column corresponding to the sensitive attribute are highlighted by a red rectangle, so we can visually see which other attribute is correlated with it, and the more a cell color tends to red or blue, the stronger is the correlation.
It can be seen that there actually are cases in which some features increase or decrease by following a certain relationship with the sensitive attribute; the clearer example is in *German*, where the feature on the third row has a correlation coefficient of -0.74 with the sensitive feature.

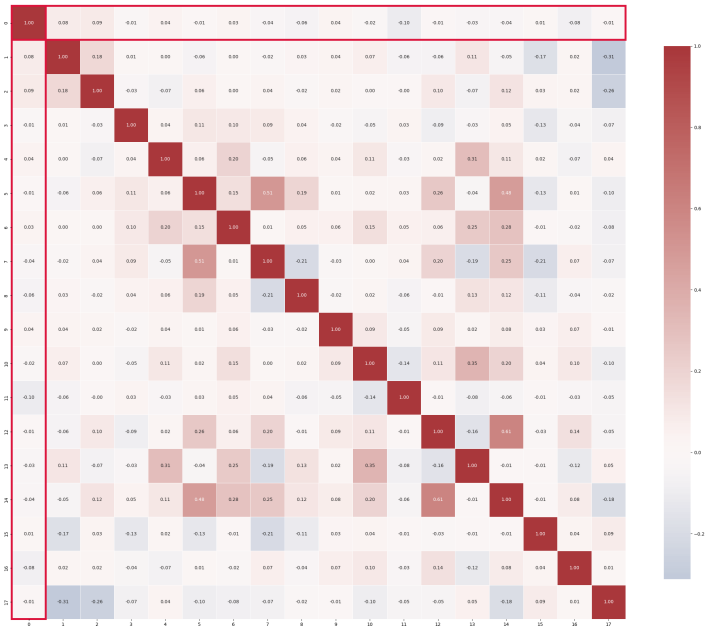


Figure 5.2: Correlation matrix for Bail dataset.

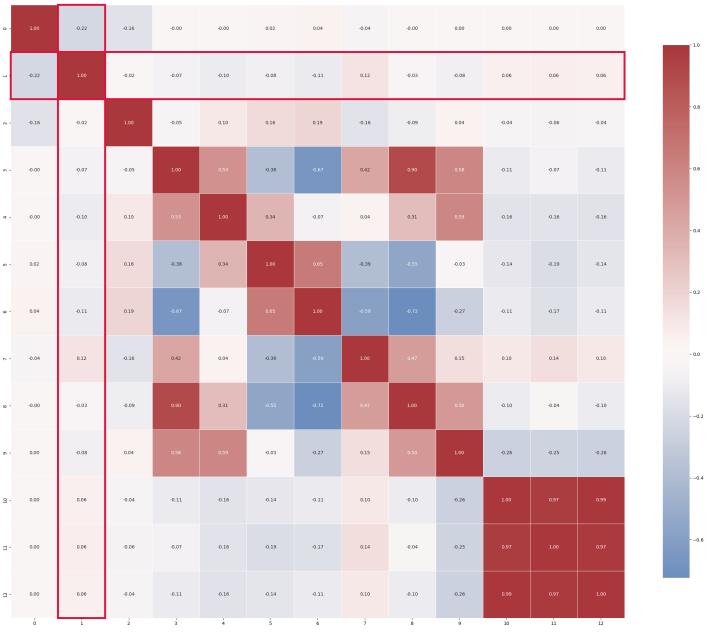


Figure 5.3: Correlation matrix for credit dataset.

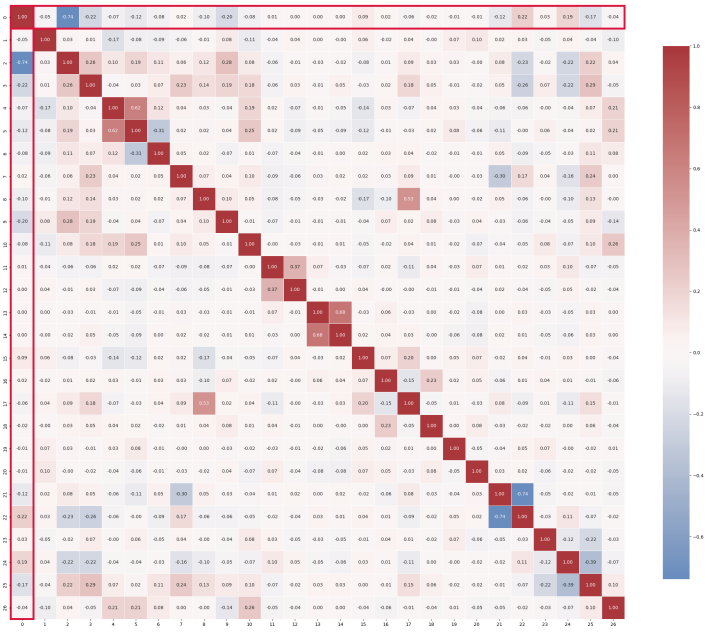


Figure 5.4: Correlation matrix for German dataset.

Before discussing the results, we should talk about the new hyperparameter introduced by adapting the idea of FairRF into NIFTY: β, that it is used to adjust the distribution of learned related features weigh λ (see eq. 3.6). Even if we do not employ the loss regularizer as in the original work, β is still implied in the weight update process (since we do not modify it) and implicitly affects the optimization process.
Another hyperparameter introduced is τ; we need it to define a minimum threshold on the correlation coefficient to select attributes most related to the sensitive one. We fix this threshold at 0.09, so that at least two features are select for each dataset (see figures 5.2, 5.3, 5.4).
We did not have much time to test many different values for β, so we report results with the ones that performed the best. We started from the values used in the original paper, i.e. $0.5, 0.8, 1$; however, those values seemed to be too high for our setting and did not bring any improvements to the baseline. Instead, lower values like 0.3 always resulted in a reduction of at least one of the fairness metrics considered.

Another hyperparameter that can influence the performances of the method is the base attribute perturbation rate r (see 3.1), which tells the probability for which every attribute will be modified; we called it dr_{attr} in the tables.

Table 5.8 shows the results with the biased attribute perturbation for German; we can see that both Δ_{SP} and Δ_{EO} are reduced by the method, especially with $D_{attr}r = 0.1$, where SP is almost $\frac{1}{3}$ and EO is less than $\frac{1}{6}$ w.r.t the results obtained by standard nifty. We can also note how, except for the SP with $D_{attr}r = 0.2$, the biased attribute perturbation seems to find solutions closer together, since they generally show less variance for the two group metrics.

Table 5.9 reports results for Bail; also in this case, we can see that both Δ_{SP} and Δ_{EO} are reduced, except for the case with $D_{attr}r = 0.2$, where Δ_{SP} is slightly increased. Another observation is that only with this dataset, AUC values are a bit higher with our extension; this is counterintuitive since in general, to reduce unfairness a model has to sacrifice some prediction accuracy (otherwise, the standard model would have found that solution). Note however that in the cases where our extension reduces both Δ_{SP} and Δ_{EO} (i.e., with $D_{attr}r = [0.01, 0.1]$), AUC values for standard NIFTY show a way higher variance that make the accuracy fall in a range of values similar to the one of our extension.

Finally, Table 5.10 shows the results for Credit; in this case, we can see that our method reduces all the fairness metrics to the same extent with every different value of $D_{attr}r$, while sacrificing a negligible amount of accuracy.

German					
$D_{attr}r$	β	AUC	Δ_{SP}	Δ_{EO}	CF
0.01		68.82 ±2.74	2.44 ±1.55	1.87 ±0.61	0.60 ±0.60
	0.3	69.53 ±2.46	1.56 ±1.19	1.19 ±0.68	0.72 ±0.81
0.1		69.78 ±2.32	3.38 ±2.48	2.85 ±2.31	0.53 ±0.55
	0.3	68.85 ±1.94	1.38 ±0.67	0.44 ±0.25	0.67 ±0.37
0.2		69.47 ±2.20	2.14 ±1.22	1.68 ±2.20	0.60 ±0.50
	0.3	69.02 ±2.75	2.08 ±1.49	1.54 ±0.77	0.47 ±0.58

Table 5.8: Comparison between NIFTY (gray rows) and biased attribute perturbation on German dataset.

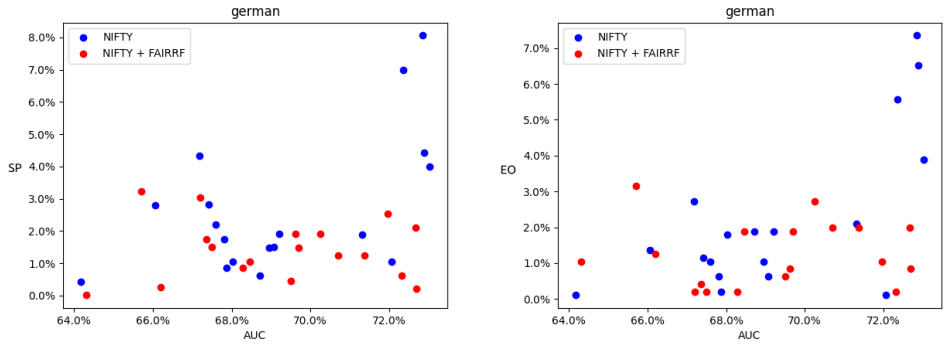
Bail					
$D_{attr}r$	β	AUC	ΔSP	ΔEO	CF
0.01		82.95 ±2.01	1.89 ±0.41	1.41 ±0.25	0.82 ±0.49
	0.3	84.28 ±0.69	1.73 ±0.27	1.09 ±0.33	1.11 ±0.18
0.1		82.51 ±2.01	1.94 ±0.76	1.22 ±0.29	1.15 ±0.32
	0.3	83.53 ±0.23	1.86 ±0.31	1.09 ±0.67	1.35 ±0.21
0.2		81.95 ±0.19	1.70 ±0.16	1.15 ±0.35	1.31 ±0.67
	0.3	84.16 ±0.27	1.88 ±0.38	0.72 ±0.29	1.20 ±0.29

Table 5.9: Comparison between NIFTY (gray rows) and and biased attribute perturbation on Bail dataset.

Credit					
$D_{attr}r$	β	AUC	ΔSP	ΔEO	CF
0.01		72.16 ±0.17	12.79 ±1.57	10.55 ±1.53	0.88 ±1.18
	0.3	72.20 ±0.18	11.69 ±0.10	9.40 ±0.11	0.06 ±0.04
0.1		72.16 ±0.17	12.85 ±1.55	10.57 ±1.51	0.90 ±1.22
	0.3	71.98 ±0.22	11.73 ±0.11	9.45 ±0.11	0.09 ±0.05
0.2		72.17 ±0.18	12.80 ±1.58	10.55 ±1.54	0.88 ±1.19
	0.3	71.83 ±0.18	11.68 ±0.04	9.40 ±0.06	0.11 ±0.09

Table 5.10: Comparison between NIFTY (gray rows) and and biased attribute perturbation on Credit dataset.

To have a more general view of the performances of our approach, we also report some images (figures 5.5, 5.6) where we plotted all the results obtained by NIFTY versus our extension (that we called in this case "NIFTY + FAIRRF"). Both figures show the little decrease in accuracy, while also decreasing the fairness metric; we see that our method's results (red points) follow the ones of the original NIFTY (blue points), while generally staying close to the bottom and a little to the left, indicating a reduction in the fairness metrics trading-off a negligible decrease in accuracy, and reflecting the values reported in the tables. It should also be noted that in both cases lowest values for the fairness metric are achieved by our implementation.



(a) **(b)**

Figure 5.5: Plotted single results for biased attribute perturbation for SP (a) and EO (b) on german dataset

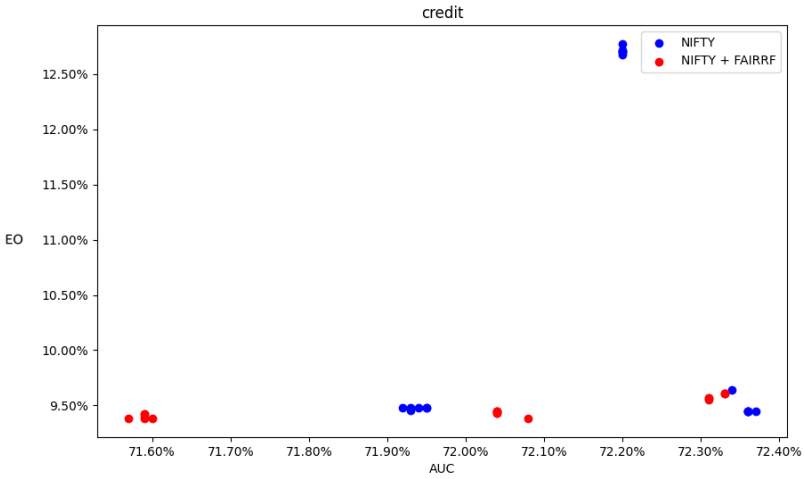


Figure 5.6: Plotted single results for biased attribute perturbation for EO on credit dataset

5.4.3 Biased edge dropout

The other extension developed, described in section 4.1, has been tested on all four datasets; since the solution proposes to modify the random perturbation process, which is concretely implemented as a random edge drop, for our experiments we select different values for the hyperparameter **edge drop rate (dr)** other than δ; the former is in fact used in NIFTY to define the probability to drop an edge when building the graph structure augmentation, while the latter specifies the ratio of homophilous edges in the set of dropped edges.
We explored different values for dr and δ: the range of values varies for every dataset involved, because of the different number of edges and level of homophily. We fixed the attribute perturbation rate (see 3.1) to 0.1 as it is the default value used in NIFTY and does not directly influences perturbation on the graph structure.

Tables 5.11, 5.12, 5.13 ad 5.14 presents our experimental results with this method. Note that with $\delta = 0$, the extension reduces to NIFTY because the original edge perturbation is random, i.e., the dropout is unbiased; in the same way, using $\delta = 0$ corresponds to dropping edges with a probability $p = 1/2$, so every edge has the same probability of being removed, independently by the fact of being homophilous or heterophilous: in other words, the perturbation is random and unbiased.

We can see that the values follow the same exact pattern seen with the biased attribute perturbation extension: even with Bail (Table 5.13) which has the more complex behavior, results are comparable, with a reduction in SP and increase in EO with $dr = 0.1$, reduction in EO and increase in SP with $dr = 0.25$. .
Finally, also in this case the smaller values for each metric are obtained by our method.

In German (Table 5.11) we see that our method improves on all the considered fairness metrics (SP, EO, and CF) compared to the baseline NIFTY (rows with $\delta = 0$), where $\delta = 0.4$ seems to provide the best results, except for the CF metric with drop rate 0.1 where we see an increase.

On Pokec (Table 5.12), we see a similar pattern for SP and CF, while EO improves compared to the baseline for drop rate 0.1 and 0.2, except with drop rate 0.2 and δ 0.49 where it slightly increase; however, it still remains near the value obtained with baseline NIFTY, but with less standard deviation. Note however that our method achieves the overall lowest value on all fairness metrics.

In Credit (Table 5.14) our method consistently improves all the fairness metrics.

In all cases, the decrease in AUC of our method compared to the baseline is always negligible, while always decreasing at least one of the fairness metrics.
From the results, it is clear that our method is a powerful extension to NIFTY, being able to significantly improve the fairness of the resulting model for all the metrics and in all the considered datasets. The choice of δ and Dr are however crucial to obtain satisfying results.
Moreover, while here we report different fairness metrics, it is a good practice in real-world applications to focus on one fairness metric and then optimize the hyperparameters for that one.

German					
Dr	δ	AUC	Δ_{SP}	Δ_{EO}	CF
0.01	0	71.05 ±1.52	2.19 ±1.57	3.12 ±0.15	0.30
	.35	71.05 ±0.57	2.15 ±1.03	3.04 ±1.11	0.20 ±0.28
	.4	70.63 ±0.38	1.34 ±0.70	2.35 ±1.06	0.43 ±0.31
0.1	0	69.78 ±2.32	3.38 ±2.48	2.85 ±2.31	0.53 ±0.55
	.35	69.45 ±1.01	2.15 ±1.49	2.36 ±1.67	0.40 ±0.28
	.4	67.17 ±1.47	0.85 ±0.57	0.81 ±0.56	0.50 ±0.41
0.2	0	68.69 ±0.91	2.26 ±1.52	2.15 ±1.52	0.87 ±0.38
	.35	68.11 ±1.1	1.58 ±1.1	1.16 ±0.69	0.47 ±0.45
	.4	68.46 ±1.35	1.19 ±0.92	1.3 ±0.96	0.47 ±0.25

Table 5.11: Comparison between NIFTY (rows with $\delta = 0$) and biased edge dropout on German dataset.

Pokec					
Dr	δ	AUC	Δ_{SP}	Δ_{EO}	CF
0.1	0	67.24 ±0.43	0.63 ±0.06	0.70 ±0.28	0.13 ±0.03
	.47	66.42 ±0.12	0.53 ±0.02	0.34 ±0.02	0.10 ±0.01
	.49	66.99 ±0.05	0.53 ±0.02	0.25 ±0.07	0.09 ±0.01
0.2	0	66.52 ±0.60	0.62 ±0.17	0.56 ±0.36	0.13 ±0.03
	.47	66.38 ±0.01	0.48 ±0.02	0.47 ±0.12	0.11 ±0.01
	.49	66.36 ±0.06	0.54 ±0.09	0.60 ±0.11	0.10 ±0.03
0.2	0	66.52 ±0.06	0.62 ±0.17	0.56 ±0.36	0.13 ±0.03
	.47	66.38 ±0.01	0.48 ±0.02	0.47±0.12	0.11 ±0.01
	.49	66.36 ±0.04	0.54 ±0.06	0.6 ±0.16	0.1 ±0.01

Table 5.12: Comparison between NIFTY (rows with $\delta = 0$) and biased edge dropout on Pokec dataset.

Figures 5.7 and 5.8 report the same kind on plots reported for the other extension and the same considerations can be done: they show a negligible reduction in accuracy while decreasing the fairness metrics.

		Bail			
Dr	δ	AUC	ΔSP	ΔEO	CF
0.1	0	81.66 ±0.08	2.26 ±0.19	0.79 ±0.36	1.04 ±0.06
	.25	82.85 ±0.37	1.40 ±0.18	1.05 ±0.10	1.37 ±0.10
	.4	83.24 ±0.14	1.67 ±0.18	1.09 ±0.25	1.21 ±0.08
0.25	0	84.28 ±0.12	2.09 ±0.18	1.11 ±0.59	1.61 ±0.20
	.25	83.51 ±0.14	2.28 ±0.18	0.61 ±0.31	1.36 ±0.09
	.4	83.97 ±0.08	2.30 ±0.15	0.44 ±0.12	1.24 ±0.07
0.3	0	83.52 ±0.06	1.97 ±0.24	1.26 ±0.21	1.17 ±0.08
	.25	83.05 ±0.76	2.22 ±0.33	0.47 ±0.15	1.4 ±0.17
	.4	83.26 ±0.11	2.28 ±0.1	0.72 ±0.23	1.18 ±0.06

Table 5.13: Comparison between NIFTY (rows with $\delta = 0$) and biased edge dropout Bail dataset.

		Credit			
Dr	δ	AUC	ΔSP	ΔEO	CF
0.01	0	72.13 ±0.01	12.66 ±0.94	10.30 ±0.88	0.78 ±0.75
	.47	72.09 ±0.01	11.72 ±0.02	9.45 ±0.02	0.08 ±0.01
	.499	72.09 ±0.02	11.73 ±0.06	9.45 ±0.05	0.10 ±0.02
0.2	0	72.15 ±0.00	12.84 ±0.02	10.54 ±0.03	0.84 ±0.01
	.47	72.06 ±0.03	11.73 ±0.05	9.45 ±0.04	0.12 ±0.03
	.499	72.04 ±0.04	11.75 ±0.03	9.46 ±0.01	0.11 ±0.03
0.25	0	72.16 ±0.00	12.69 ±0.01	10.35 ±0.13	0.68 ±0.09
	.47	72.13 ±0.02	11.73 ±0.02	9.46 ±0.02	0.04 ±0.02
	.49	72.09 ±0.03	11.73 ±0.02	9.46 ±0.02	0.07 ±0.01

Table 5.14: Comparison between NIFTY (rows with $\delta = 0$) and biased edge dropout on Credit dataset.

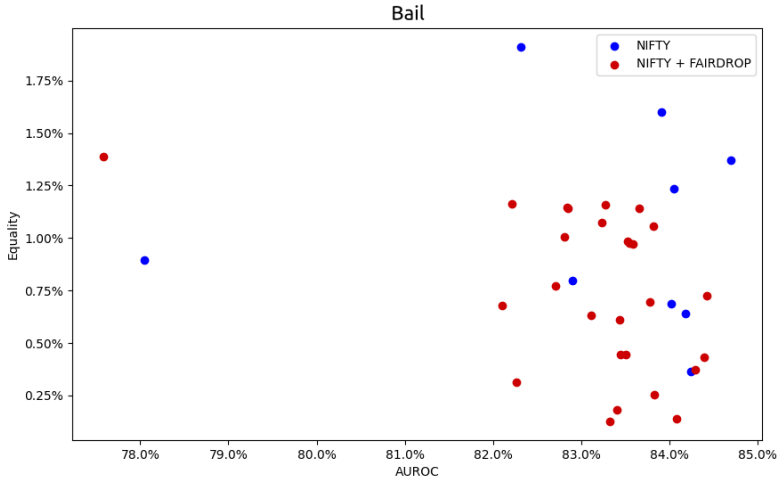


Figure 5.7: Plotted single results for biased edge perturbation EO on bail dataset

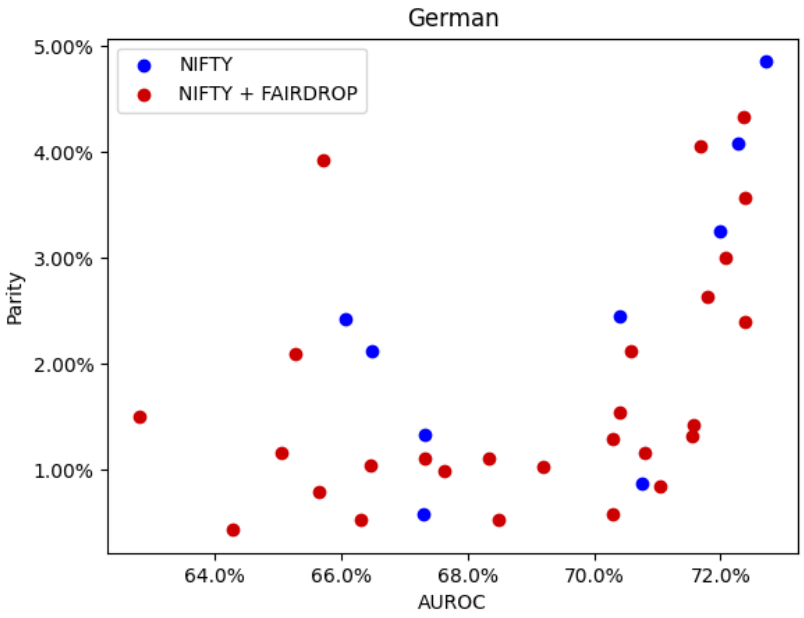


Figure 5.8: Plotted single results for biased edge perturbation SP on German dataset

In general, results show a lower difference with the baseline NIFTY w.r.t. the biased attribute perturbation extension: this may indicate that node's attributes have a major role in introducing unfairness in the learned representation.

5.4.4 Combining the methods

Since both extensions provide good results, our last experiment consists in combining them, to see if working on both attributes and graph structure can actually work better than the single methods.

Tables 5.15, 5.16 and 5.17 show the results obtained combining the two methods for German, Credit and Bail. Note that the hyperparameter β is not reported in the table, but we only used the value 0.3; we instead report both Dr and $D_{attr}r$ (edge and attribute perturbation rate, respectively, see 3.1 along with δ (see 4.1).

From the results, we can see that combining both methods reduces the fairness metrics while sacrificing the same level of accuracy of the extensions taken independently. Not always the combination provides the best results; sometimes it produces solutions with values that are more or less the mean between the values of the independent extensions and sometimes with worse values.

We can compare the results obtained with the same values for the hyperparameters Dr and $D_{attr}r$ in the two extensions: for example, with German, we take the results obtained with $D_{attr}r = 0.1$ and $Dr = 0.1$; the biased attribute perturbation extension has $\Delta_{SP} = 1.38$ and $\Delta_{EO} = 0.44$, while the biased edge perturbation has $\Delta_{SP} = 2.15$ and $\Delta_{EO} = 2.36$ with $delta = 0.35$. The combination has $\Delta_{SP} = 1.00$ and $\Delta_{EO} = 1.61$, so it decrease SP w.r.t. to both independent methods, while decreasing EO only w.r.t. to the biased edge perturbation with a value that is a bit lower than the mean of the two. This is however not true if we take $\delta = 0.4$, where the values for both metrics are higher than the values obtained with the two extensions, while staying lower w.r.t. the original NIFTY model.

Following the same reasoning, we see that with Bail the combination seems to have results with almost always lower values for the fairness metrics w.r.t. the independent extensions; with Credit, the combination provides results that are very similar to the biased attribute perturbation and slightly better (i.e., lower fairness metrics) w.r.t. the results of biased edge dropout.
In any case, the combination of the two methods improves in terms of fairness w.r.t. the baseline NIFTY and in particular it looks like using both methods together tends to obtain low for the counterfactual fairness metric.

A more accurate selection for the hyperparameter β should be done, to make biased attribute perturbation affect more or less the model; there are also a lot of combinations for the perturbation rates values to be tested, so it may be possible to find solutions that take the best from both extensions, but we did not have enough time to explore many of them.

German						
$D_{attr}r$	Dr	δ	AUC	Δ_{SP}	Δ_{EO}	CF
0.01	0.1	0	68.82 ±2.74	2.44 ±1.55	1.87 ±0.61	0.60 ±0.60
		0.35	68.15 ±3.43	1.00 ±1.50	1.61 ±1.51	0.66 ±0.59
		0.4	68.62 ±2.83	2.06 ±1.77	2.25 ±2.20	0.13 ±0.30
0.1		0	69.78 ±2.32	3.38 ±2.48	2.85 ±2.31	0.53 ±0.55
		0.35	66.76 ±1.54	0.99 ±0.70	0.78 ±0.79	0.31 ±0.41
		0.4	68.32 ±2.33	1.17 ±1.08	0.63 ±0.38	0.33 ±0.43

Table 5.15: Comparison between NIFTY (rows with $\delta = 0$) and the combination of our methods on German dataset.

Credit						
$D_{attr}r$	Dr	δ	AUC	Δ_{SP}	Δ_{EO}	CF
0.01	0.25	0	72.13 ±0.01	12.66 ±0.94	10.30 ±0.88	0.78 ±0.75
		0.47	72.19 ±0.15	11.67 ±0.11	9.39 ±0.11	0.06 ±0.04
		0.499	71.89 ±0.37	11.70 ±0.10	9.42 ±0.10	0.11 ±0.02
0.1		0	72.16±0.17	12.85±1.55	10.58±1.51	0.90±1.22
		0.47	71.97 ±0.23	11.74 ±0.09	9.46 ±0.09	0.07 ±0.05
		0.499	71.98 ±0.22	11.71 ±0.09	9.43 ±0.10	0.06 ±0.04
0.2		0	72.17 ±0.18	12.79 ±1.58	10.54 ±1.54	0.88 ±1.19
		0.47	71.90 ±0.21	11.69 ±0.05	9.44 ±0.06	0.10 ±0.07
		0.499	71.89 ±0.22	11.67 ±0.06	9.42 ±0.05	0.13 ±0.1

Table 5.16: Comparison between NIFTY (rows with $\delta = 0$) and the combination of our methods on Credit dataset.

Bail						
$D_{attr}r$	Dr	δ	AUC	Δ_{SP}	Δ_{EO}	CF
0.1	0.1	0	82.51 ±2.01	1.94 ±0.76	0.72 ±0.29	1.15 ±0.32
		0.25	83.70 ±0.80	1.51 ±0.15	1.07 ±0.45	1.23 ±0.18
		0.4	83.59 ±0.72	1.54 ±0.28	1.39 ±0.23	0.14 ±0.15
0.2		0	81.95 ±0.19	1.70 ±0.16	1.15 ±0.35	1.31 ±0.67
		0.25	83.09 ±0.87	1.69 ±0.42	1.01 ±0.60	1.31 ±0.41
		0.4	82.97 ±0.76	1.71 ±0.42	0.98 ±0.75	1.37 ±0.36

Table 5.17: Comparison between NIFTY (rows with $\delta = 0$) and the combination of our methods on Bail dataset.

Chapter 6

Conclusions and future works

In this thesis, we studied and proposed two extensions to improve fairness in the state-of-the-art model NIFTY, a method to compute fair node representations.
We replaced the randomness in its perturbation processes to create augmented views that force the learned representation to give less importance to sensitive and sensitive-related features instead of introducing random noise.

Our first proposal is inspired by recently proposed biased graph structure modification methods; our second proposal exploits an idea taken from a recent work on fair classification and adapts it as a node attributes perturbation strategy.
We have shown the effectiveness of our approaches in four real-world graph datasets.

This work provides many possibilities for future extensions and refinements, also because of the lack of time that did not permit to evaluate many values for the hyperparameters used. Some possible ideas:

1. An obvious and straightforward extension would be to better tune the parameters of the backbone GCN, maybe also by trying other convolutions described in the literature;

2. Concerning the graph structure, since German, Bail, and Credit datasets have all "artificial" edges, a first idea can be to modify the link creation process by employing different similarity metrics, which could probably also be learned by a DL model from other real-world datasets of the same topic; another idea could be to inject some fairness constraints while generating edges, to obtain a fairer adjacency matrix.

3. A challenging while interesting idea is to extend this work to operate also in a context where the sensitive attributes are more than one; for example, in German dataset, there are other features that could be considered sensible, like Age or the fact of coming from another country. It can however be difficult because it also introduces the problem of *Gerrymandering* [38], which can arise when a classifier is fair for each individual group but discriminates on subgroups, such as certain combinations of different sensitive attributes.

4. Following the work of MccNifty [78] and [33], one could try to include subgraph information by adding a *"subgraph perturbation"* process, where subgraphs are generated or simply taken from nodes far from each other, with different sensitive

attribute but same ground truth, to make the model learn that the representation of the two subgraphs should be similar.

5. An idea to include *individual fairness* metrics could be to extend FairDrop, where instead of only dropping edges we replace some of them (maybe with the same heuristics used for edge generation in NIFTY) while preserving the high-order structure,i.e., remove some direct connection between homophilous edges and replace them with a path of length > 1 or create a shorter path to nodes that are connected with a long path and have different sensitive attribute while sharing most of the other features. The idea is to build an adjacency matrix where each node has more or less the same number of homophilous and heterophilous edges: in this case, optimizing for the individual metric *consistency* can also guarantee group fairness because it forces the model to do similar prediction to neighbors nodes.

6. An extension for the biased attribute perturbation process could involve the actual perturbation value other than the probability of doing it: weights (see 4.2) could be used to increase or decrease the value of the augmentation d (see 3.1). In this case, larger weights would produce a larger variation w.r.t. related attributes' values in the original node.

Bibliography

[1] C. Agarwal, H. Lakkaraju, and M. Zitnik. "Towards a Unified Framework for Fair and Stable Graph Representation Learning". In: *UAI*. 2021 (cit. on pp. 3, 15, 28, 33).

[2] Amr Ahmed et al. "Distributed Large-scale Natural Graph Factorization". In: *IW3C2 - International World Wide Web Conference*. Rio de Janeiro, Brazil, May 2013, p. 37. DOI: 10.1145/2488388.2488393. URL: https://hal.archives-ouvertes.fr/hal-00918478 (cit. on p. 12).

[3] Solon Barocas, Moritz Hardt, and Arvind Narayanan. *Fairness and Machine Learning*. http://www.fairmlbook.org. fairmlbook.org, 2019 (cit. on p. 21).

[4] Mikhail Belkin and Partha Niyogi. "Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering". In: *Advances in Neural Information Processing Systems*. Ed. by T. Dietterich, S. Becker, and Z. Ghahramani. Vol. 14. MIT Press, 2001. URL: https://proceedings.neurips.cc/paper/2001/file/f106b7f99d2cb30c3db1c3cc0fde9ccb-Paper.pdf (cit. on p. 12).

[5] Reuben Binns. "On the Apparent Conflict Between Individual and Group Fairness". In: *CoRR* abs/1912.06883 (2019). arXiv: 1912.06883. URL: http://arxiv.org/abs/1912.06883 (cit. on p. 25).

[6] A. J. Bose and W. L. Hamilton. "Compositional Fairness Constraints for Graph Embeddings". In: *arXiv* (2019) (cit. on pp. 21, 27).

[7] A. Buyl and T. De Bie. "DeBayes: a Bayesian method for debiasing network embeddings". In: *arXiv* (2020) (cit. on pp. 21, 31).

[8] Shaosheng Cao, Wei Lu, and Qiongkai Xu. "GraRep: Learning Graph Representations with Global Structural Information". In: *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. CIKM '15. Melbourne, Australia: Association for Computing Machinery, 2015, pp. 891–900. ISBN: 9781450337946. DOI: 10.1145/2806416.2806512. URL: https://doi.org/10.1145/2806416.2806512 (cit. on p. 12).

[9] Siheng Chen et al. "Discrete Signal Processing on Graphs: Sampling Theory". In: *CoRR* abs/1503.05432 (2015). arXiv: 1503.05432. URL: http://arxiv.org/abs/1503.05432 (cit. on p. 17).

[10] Alexandra Chouldechova. *Fair prediction with disparate impact: A study of bias in recidivism prediction instruments*. 2016. DOI: 10.48550/ARXIV.1610.07524. URL: https://arxiv.org/abs/1610.07524.

[11] Alexandra Chouldechova and Aaron Roth. "A Snapshot of the Frontiers of Fairness in Machine Learning". In: *Commun. ACM* 63.5 (Apr. 2020), pp. 82–89. ISSN: 0001-0782. DOI: 10.1145/3376898. URL: https://doi.org/10.1145/3376898 (cit. on p. 20).

[12] S. Dai and S. Wang. "FairGNN: Eliminating the Discrimination in Graph Neural Networks with Limited Sensitive Attribute Information". In: *arXiv* (2020) (cit. on pp. 21, 27, 39, 40).

[13] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering". In: *CoRR* abs/1606.09375 (2016). arXiv: 1606.09375. URL: http://arxiv.org/abs/1606.09375 (cit. on pp. 17, 18).

[14] Y. Dong et al. "EDITS: Modeling and Mitigating Data Bias for Graph Neural Networks". In: (2021) (cit. on p. 31).

[15] Y. Dong et al. "Individual Fairness for Graph Neural Networks: A Ranking Based Approach". In: *ACM SIGKDD*. New York, NY, USA, 2021 (cit. on p. 31).

[16] Cynthia Dwork et al. "Fairness Through Awareness". In: *CoRR* abs/1104.3913 (2011). arXiv: 1104.3913. URL: http://arxiv.org/abs/1104.3913 (cit. on pp. 23, 25).

[17] Will Fleisher. "What's Fair about Individual Fairness?" In: *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*. New York, NY, USA: Association for Computing Machinery, 2021, pp. 480–490. ISBN: 9781450384735. URL: https://doi.org/10.1145/3461702.3462621 (cit. on p. 24).

[18] Luciano Floridi and Josh Cowls. "A Unified Framework of Five Principles for AI in Society". In: *Harvard Data Science Review* 1.1 (July 2019) (cit. on p. 18).

[19] M. Gori, G. Monfardini, and F. Scarselli. "A new model for learning in graph domains". In: *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* Vol. 2. 2005, 729–734 vol. 2. DOI: 10.1109/IJCNN.2005.1555942 (cit. on p. 16).

[20] Aditya Grover and Jure Leskovec. "node2vec: Scalable Feature Learning for Networks". In: *CoRR* abs/1607.00653 (2016). arXiv: 1607.00653. URL: http://arxiv.org/abs/1607.00653 (cit. on pp. 12, 27).

[21] Yosh Halberstam and Brian Knight. "Homophily, group size, and the diffusion of political information in social networks: Evidence from Twitter". In: *Journal of Public Economics* 143 (2016), pp. 73–88. ISSN: 0047-2727. DOI: https://doi.org/10.1016/j.jpubeco.2016.08.011. URL: https://www.sciencedirect.com/science/article/pii/S0047272716301001 (cit. on p. 29).

[22] Will Hamilton, Zhitao Ying, and Jure Leskovec. "Inductive Representation Learning on Large Graphs". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017 (cit. on p. 18).

[23] William L. Hamilton, Rex Ying, and Jure Leskovec. "Representation Learning on Graphs: Methods and Applications". In: *CoRR* abs/1709.05584 (2017). arXiv: 1709.05584. URL: http://arxiv.org/abs/1709.05584 (cit. on p. 12).

[24] M. Hardt, E. Price, and N. Srebro. "Equality of Opportunity in Supervised Learning". In: *NeurIPS*. 2016.

[25] Moritz Hardt, Eric Price, and Nathan Srebro. "Equality of Opportunity in Supervised Learning". In: *CoRR* abs/1610.02413 (2016). arXiv: 1610.02413. URL: http://arxiv.org/abs/1610.02413 (cit. on p. 22).

[26] Corinna Hertweck, Christoph Heitz, and Michele Loi. "On the Moral Justification of Statistical Parity". In: *CoRR* abs/2011.02079 (2020). arXiv: 2011.02079. URL: https://arxiv.org/abs/2011.02079 (cit. on p. 21).

[27] Peter D Hoff, Adrian E Raftery, and Mark S Handcock. "Latent Space Approaches to Social Network Analysis". In: *Journal of the American Statistical Association* 97.460 (2002), pp. 1090–1098. DOI: 10.1198/016214502388618906. eprint: https://doi.org/10.1198/016214502388618906. URL: https://doi.org/10.1198/016214502388618906 (cit. on p. 12).

[28] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5 (1989), pp. 359–366. ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(89)90020-8. URL: https://www.sciencedirect.com/science/article/pii/0893608089900208 (cit. on p. 9).

[29] *https://colab.research.google.com/*. URL: https://colab.research.google.com/ (cit. on p. 40).

[30] *https://github.com/Fecsio*. URL: https://github.com/Fecsio (cit. on p. 41).

[31] Zexi Huang, Arlei Silva, and Ambuj K. Singh. "A Broader Picture of Random-walk Based Graph Embedding". In: *CoRR* abs/2110.12344 (2021). arXiv: 2110.12344. URL: https://arxiv.org/abs/2110.12344 (cit. on p. 12).

[32] Daniel T. Jones James P. Womack. *Lean Thinking, Second Editon*. Simon & Schuster, Inc., 2010.

[33] Yizhu Jiao et al. "Sub-graph Contrast for Scalable Self-Supervised Graph Representation Learning". In: *CoRR* abs/2009.10273 (2020). arXiv: 2009.10273. URL: https://arxiv.org/abs/2009.10273 (cit. on p. 59).

[34] Toshihiro Kamishima, Shotaro Akaho, and Jun Sakuma. "Fairness-aware Learning through Regularization Approach". In: *2011 IEEE 11th International Conference on Data Mining Workshops*. 2011, pp. 643–650. DOI: 10.1109/ICDMW.2011.83 (cit. on p. 27).

[35] Toshihiro Kamishima, Shotaro Akaho, and Jun Sakuma. "Fairness-aware learning through regularization approach". In: *2011 IEEE 11th International Conference on Data Mining Workshops*. IEEE. 2011, pp. 643–650 (cit. on p. 27).

[36] B. Kang, J. Lijffijt, and T. De Bie. "Conditional Network Embeddings". In: *ICLR*. 2019 (cit. on p. 31).

[37] Michael Kearns, Aaron Roth, and Zhiwei Steven Wu. "Meritocratic Fairness for Cross-Population Selection". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 1828–1836. URL: https://proceedings.mlr.press/v70/kearns17a.html (cit. on p. 24).

[38] Michael Kearns et al. "Preventing Fairness Gerrymandering: Auditing and Learning for Subgroup Fairness". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 2564–2572. URL: https://proceedings.mlr.press/v80/kearns18a.html (cit. on p. 59).

[39] Moein Khajehnejad et al. "Adversarial Graph Embeddings for Fair Influence Maximization over Social Networks". In: *ArXiv* abs/2005.04074 (2020) (cit. on p. 31).

[40] T. N. Kipf and M. Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *ICLR*. 2017.

[41] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *CoRR* abs/1609.02907 (2016). arXiv: 1609.02907. URL: http://arxiv.org/abs/1609.02907 (cit. on p. 18).

[42] Nikita Kozodoi, Johannes Jacob, and Stefan Lessmann. "Fairness in credit scoring: Assessment, implementation and profit implications". In: *European Journal of Operational Research* 297.3 (2022), pp. 1083–1094. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2021.06.023. URL: https://www.sciencedirect.com/science/article/pii/S0377221721005385 (cit. on p. 27).

[43] Matt J. Kusner et al. *Counterfactual Fairness*. 2017. DOI: 10.48550/ARXIV.1703.06856. URL: https://arxiv.org/abs/1703.06856 (cit. on p. 28).

[44] C. Laclau et al. "All of the Fairness for Edge Prediction with Optimal Transport". In: *arXiv* (2020).

[45] Anja Lambrecht and Catherine Tucker. "Algorithmic Bias? An Empirical Study of Apparent Gender-Based Discrimination in the Display of STEM Career Ads". In: *Management Science* 65.7 (2019), pp. 2966–2981. DOI: 10.1287/mnsc.2018.3093. URL: https://doi.org/10.1287/mnsc.2018.3093 (cit. on p. 19).

[46] Michelle Lee, Luciano Floridi, and Jat Singh. "Formalising trade-offs beyond algorithmic fairness: lessons from ethical philosophy and welfare economics". In: *AI and Ethics* 1 (Nov. 2021). DOI: 10.1007/s43681-021-00067-y (cit. on p. 21).

[47] O. Li et al. "On Dyadic Fairness: Exploring and Mitigating Bias in Graph Connections". In: *ICLR*. 2021 (cit. on p. 31).

[48] Daniel Lowd and Christopher Meek. "Adversarial Learning". In: *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*. KDD '05. Chicago, Illinois, USA: Association for Computing Machinery, 2005, pp. 641–647. ISBN: 159593135X. DOI: 10.1145/1081870.1081950. URL: https://doi.org/10.1145/1081870.1081950 (cit. on p. 27).

[49] Jing Ma et al. "Learning Fair Node Representations with Graph Counterfactual Fairness". In: *CoRR* abs/2201.03662 (2022). arXiv: 2201.03662. URL: https://arxiv.org/abs/2201.03662 (cit. on p. 27).

[50] Warren S. McCulloch and Walter Pitts. "A Logical Calculus of the Ideas Immanent in Nervous Activity". In: *The Bulletin of Mathematical Biophysics* 5.4 (1943), pp. 115–133. DOI: 10.1007/bf02478259 (cit. on p. 7).

[51] Ninareh Mehrabi et al. "A Survey on Bias and Fairness in Machine Learning". In: *ACM Comput. Surv.* 54.6 (July 2021). ISSN: 0360-0300. DOI: 10.1145/3457607. URL: https://doi.org/10.1145/3457607 (cit. on p. 25).

[52] A. Micheli. "Neural network for graphs: A contextual constructive approach". In: *IEEE Transactions on Neural Networks* 20.3 (2009), pp. 498–511.

[53] Alessio Micheli. "Neural Network for Graphs: A Contextual Constructive Approach". In: *IEEE Transactions on Neural Networks* 20.3 (2009), pp. 498–511. DOI: 10.1109/TNN.2008.2010350 (cit. on p. 16).

[54] N. Navarin, L. Oneto, and M. Donini. "Learning Deep Fair Graph Neural Networks". In: *ESANN*. 2020.

[55] Nicolo Navarin et al. "Linear graph convolutional networks". In: *ESANN*. 2020 (cit. on p. 17).

[56] Luca Oneto and Silvia Chiappa. "Fairness in Machine Learning". In: *CoRR* abs/2012.15816 (2020). arXiv: 2012.15816. URL: https://arxiv.org/abs/2012.15816 (cit. on p. 25).

[57] Lawrence Page et al. *The PageRank Citation Ranking: Bringing Order to the Web.* Technical Report 1999-66. Previous number = SIDL-WP-1999-0120. Stanford InfoLab, Nov. 1999. URL: http://ilpubs.stanford.edu:8090/422/ (cit. on p. 31).

[58] *Proposal for a Regulation laying down harmonised rules on artificial intelligence.* URL: https://digital-strategy.ec.europa.eu/en/library/proposal-regulation-laying-down-harmonised-rules-artificial-intelligence (cit. on p. 19).

[59] *PyTorch.* URL: https://pytorch.org/ (cit. on p. 40).

[60] *PyTorch-Geometric.* URL: https://www.pyg.org/ (cit. on p. 41).

[61] Tahleen Rahman et al. "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, July 2019, pp. 3289–3295. DOI: 10.24963/ijcai.2019/456. URL: https://doi.org/10.24963/ijcai.2019/456 (cit. on p. 27).

[62] Y. Rong et al. "DropEdge: Towards Deep Graph Convolutional Networks on Node Classification". In: *ICLR*. 2020.

[63] F. Rosenblatt. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms.* Cornell Aeronautical Laboratory. Report no. VG-1196-G-8. Spartan Books, 1962. URL: https://books.google.it/books?id=7FhRAAAAMAAJ (cit. on p. 7).

[64] Ahmed Ali Mohammed Al-Saffar, Hai Tao, and Mohammed Ahmed Talab. "Review of deep convolution neural network in image classification". In: *2017 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET)*. 2017, pp. 26–31. DOI: 10.1109/ICRAMET.2017.8253139 (cit. on p. 15).

[65] F. Scarselli et al. "The Graph Neural Network Model". In: *IEEE Transactions on Neural Networks* 20.1 (2009), pp. 61–80.

[66] Franco Scarselli et al. "The Graph Neural Network Model". In: *IEEE Transactions on Neural Networks* 20.1 (2009), pp. 61–80. DOI: 10.1109/TNN.2008.2005605 (cit. on p. 16).

[67] David I Shuman et al. "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains". In: *IEEE Signal Processing Magazine* 30.3 (2013), pp. 83–98. DOI: 10.1109/MSP.2012.2235192 (cit. on p. 17).

[68] A. Sperduti and A. Starita. "Supervised neural networks for the classification of structures". In: *IEEE Transactions on Neural Networks* 8.3 (1997), pp. 714–735.

[69] Alessandro Sperduti, Darya Majidi, and Antonina Starita. "Extended Cascade-Correlation for syntactic and structural pattern recognition". In: *Advances in Structural and Syntactical Pattern Recognition*. Ed. by Petra Perner, Patrick Wang, and Azriel Rosenfeld. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 90–99. ISBN: 978-3-540-70631-1 (cit. on p. 16).

[70] I. Spinelli et al. "FairDrop: Biased Edge Dropout for Enhancing Fairness in Graph Representation Learning". In: *IEEE Transactions on Artificial Intelligence* (2021) (cit. on pp. 3, 28, 29, 33, 34).

[71] Alan Tsang et al. "Group-Fairness in Influence Maximization". In: *CoRR* abs/1903.00967 (2019). arXiv: 1903.00967. URL: http://arxiv.org/abs/1903.00967 (cit. on p. 31).

[72] Petar Velickovic et al. "Graph attention networks". In: *stat* 1050 (2017), p. 20 (cit. on p. 18).

[73] Sahil Verma and Julia Rubin. "Fairness Definitions Explained". In: *Proceedings of the International Workshop on Software Fairness*. FairWare '18. Gothenburg, Sweden: Association for Computing Machinery, 2018, pp. 1–7. ISBN: 9781450357463. DOI: 10.1145/3194770.3194776. URL: https://doi.org/10.1145/3194770.3194776 (cit. on p. 22).

[74] Yanhao Wei et al. "Credit scoring with social network data". In: *Marketing Science* 35.2 (2016), pp. 234–258 (cit. on p. 27).

[75] Zonghan Wu et al. "A Comprehensive Survey on Graph Neural Networks". In: *CoRR* abs/1901.00596 (2019). arXiv: 1901.00596. URL: http://arxiv.org/abs/1901.00596 (cit. on p. 17).

[76] Keyulu Xu et al. "How Powerful are Graph Neural Networks?" In: *CoRR* abs/1810.00826 (2018). arXiv: 1810.00826. URL: http://arxiv.org/abs/1810.00826 (cit. on p. 18).

[77] Ziqian Zeng et al. "Fair Representation Learning for Heterogeneous Information Networks". In: *CoRR* abs/2104.08769 (2021). arXiv: 2104.08769. URL: https://arxiv.org/abs/2104.08769 (cit. on p. 21).

[78] Xu Zhang et al. "A Multi-view Confidence-calibrated Framework for Fair and Stable Graph Representation Learning". In: *2021 IEEE International Conference on Data Mining (ICDM)*. 2021, pp. 1493–1498. DOI: 10.1109/ICDM51629.2021.00194 (cit. on pp. 31, 59).

[79] Tianxiang Zhao et al. "You Can Still Achieve Fairness Without Sensitive Attributes: Exploring Biases in Non-Sensitive Features". In: *CoRR* abs/2104.14537 (2021). arXiv: 2104.14537. URL: https://arxiv.org/abs/2104.14537 (cit. on pp. 3, 21, 28, 30, 33, 36).