Ecole Polytechnique Fédérale de Lausanne

Laboratory for Information and Inference Systems

Semester Project : Blind Fairness

Author:
Pablo RUIZ DE VELASCO 261406

Supervisors:
Dr. Grigorios Chrysos
Prof. Cevher Volkan

July 13, 2022

Contents

1 Introduction

This project is a study in the field of fairness in machine learning (ML). This field has recently emerged over the last few years as ML researchers realized that having a high performing model that is fair is one of the most important aspects of a model for it to be used in real world applications. The notion of fairness arises from the fact that many datasets used for training models contain biases. For instance, a picture of a cow will often have a green background because most of the pictures of cows are taken in grass fields. However, this doesn't mean that every animal on a green background is a cow, or that a picture of a cow will never have a background color other than green. Unfortunately, ML models do pick up on these spurious correlations to take their decisions, making them unfair to subgroups poorly represented on the training data. This in turn makes the model unusable for real world applications with high stakes such as credit loan assignment or employee recruitment.

Many methods have recently been developed to ensure fairness in ML models. Fairness Through Unawareness (FTU, a.k.a. Blind Fairness) is a subset of these methods that requires the data to be annotated with a subgroup label so that this information can be used at training time by the model to ensure fairness. The main drawback of this approach is that collecting the subgroup labels can be expensive and time-consuming, or unfeasible due to privacy reasons (e.g. attributes such as gender in the case of samples corresponding to persons). In this project, we therefore focus on finding an approach to recover the subgroup labels from a dataset so that they can be later used by models implementing Blind Fairness.

Section 2 prepares the reader with a few notations and fairness notions that will be used throughout this report. Section 3 focuses on the related work, while section 4 describes the approach taken to achieve the desired goal. Section 5 presents the results obtained on different experiments, and section 6 concludes this report.

2 Notations and Fairness Notions

2.1 Notations

To ease the reading of this report, we first establish a few notations.
As usual in supervised ML, we will always have a dataset composed of N labelled samples $\{x_i, y_i\}_{i=1}^N$, where $x_i \in \mathbb{R}^d$ corresponds to a sample, and $y_i \in \{0, ..., n-1\}$ corresponds to its label, assuming n classes. When we won't be referring to a specific realisation of a labelled sample, we will use capital letters to denote the corresponding random variables, i.e. X for the samples and Y for the labels. In the context of fairness, we will also have a subgroup label $a_i \in \{0, ..., m-1\}$ indicating for each sample x_i to which subgroup of the sample population it belongs, assuming m subgroups. The notion of subgroups in the data will be more clearly explained in the following section. In addition, $\hat{Y}$, $\hat{A}$ (or $\hat{y}_i$, $\hat{a}_i$ for a specific realisation) will always respectively denote the label and subgroup predictions made by a ML model.

2.2 Fairness Notions

In this work, we define subgroups in the data as follows. We assume the data contains a bias which can appear for different reasons, a very frequent one being selection bias, where the data is not representative of the true data distribution because of how it was collected. This bias translates into a feature, that would normally be irrelevant to the classification task, being correlated with the class labels. We therefore call this correlation a spurious correlation, and refer to the associated feature as a spurious feature. Throughout this report, we define subgroups as sets of samples sharing the same spurious feature. If we have a single spurious feature taking m different values, then we have m subgroups present in the data. If we have more than one spurious feature, we can consider an artificial one taking as many values as there are combinations of the initial spurious features. For instance, if in the data we have gender (male, female) and hair color (blond, brown, red) as spurious features, then we have six subgroups e.g. blond males, blond females etc.

There are mainly two settings in which ML researchers study fairness, Blind Fairness (BF) and Blind Fairness without Demographics. The key difference between them is that in the case of the former, the dataset being used contains subgroup labels indicating the membership of a sample to a specific subgroup, whereas in the case of the latter, these subgroup labels are unknown, making it harder to

ensure fairness. The name Blind Fairness or FTU comes from the idea that a classifier will be fair to the different subgroups if the learned sample representation is independent of the subgroup, i.e. the classifier will be unaware of the subgroups present in the data.

2.3 Common Fairness Metrics

There are several fairness metrics that have been developed to evaluate the fairness of a model. Here we focus on the family of group-fairness metrics, which reason on the model at the scale of the subgroups present in the data.

First, we have the *demographic parity*, which states that a fair model should satisfy the following equality, where $\hat{Y}$ are the predictions of the model and A is the subgroup label:

$$P(\hat{Y} = y|A = a_k) = P(\hat{Y} = y|A = a_l) \quad \forall y \in \mathcal{Y},\ \forall a_k, a_l \in \mathcal{A} \tag{1}$$

This metric simply states that the probability of the model predicting $\hat{Y} = y$ should be independent of the subgroup label A. It should be noted that this metric assumes that the distribution of the ground truth labels is the same on the subgroups, i.e. $P(Y = y|A = a_k) = P(Y = y|A = a_l)$, which is not always the case.

Next, we have the *equalized odds* metric, which requires the following equality to be satisfied, where Y represents the ground truth labels:

$$P(\hat{Y} \neq Y|A = a_k, Y = y) = P(\hat{Y} \neq Y|A = a_l, Y = y) \quad \forall y \in \mathcal{Y},\ \forall a_k, a_l \in \mathcal{A} \tag{2}$$

Compared to demographic parity, equalized odds requires the sample representation to be independent of the subgroup conditioned on the ground truth class labels. A model satisfying equalized odds therefore has equal error rates across subgroups and classes. There exists a relaxed version of equalized odds for the binary case called *equal opportunity* where only the case $y = 1$ is considered.

3 Related Work

In this section, we review papers from the fairness literature and present different approaches aiming at obtaining fair models that have been shown to work.

3.1 Blind Fairness

In the context of Blind Fairness, we consider two papers [1] and [2] respectively authored by David Madras et al. and Byungju Kim et al. Both papers propose similar solutions for the FTU problem. We will also refer to the papers by the name of their implementation which is Learning Fair and Transferable Representations (LAFTR) for [1] and Learning Not To Learn (LNTL) for [2].

In [1], the authors suggest a way of learning fair representations, i.e. sample representations that can be used by a model for classification tasks and that are fair with respect to a sensitive attribute / spurious feature. The latter can be understood as obtaining representations not encoding any information about the sensitive attribute, i.e. representations independent of the sensitive attribute. Their framework consists of a binary classification task with samples having a binary sensitive attribute, but it can be generalized to multi-class classification problems with sensitive attributes taking more than two values. They propose the architecture depicted in Fig. 1. In this architecture, the encoder f learns representations Z from the data samples X (and optionally using their corresponding sensitive attributes A). The classifier g uses these representations for the classification task while the adversary h uses them to try to predict the sensitive attribute A. The decoder module is optional, its inclusion in the architecture is not crucial for learning fair representations.

This architecture results in the following optimisation problem:

$$\min_{f,g,k} \max_{h} \mathbb{E}_{X,Y,A}\left[L(f, g, h, k)\right] \tag{3}$$

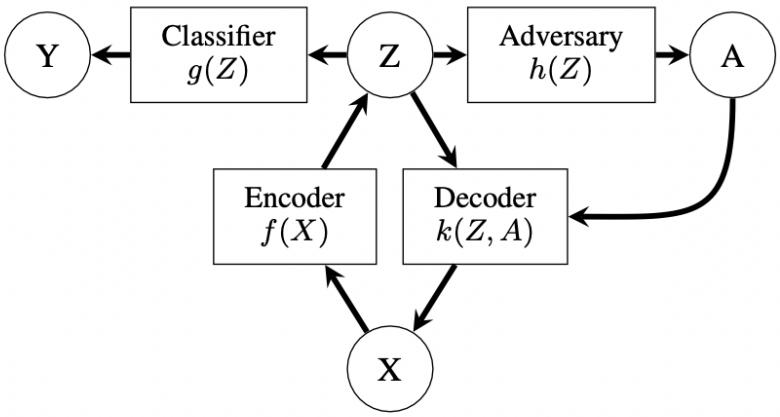

Figure 1: LAFTR model.

The objective function $L(f, g, h, k)$ consists of three terms, L_C for the classification task, L_{Dec} for the input reconstruction task and L_{Adv} for the sensitive attribute prediction, each of them with their respective weights α, β and γ.

$$L(f, g, h, k) = \alpha L_C\left(g(f(X, A)), Y\right) + \beta L_{Dec}\left(k(f(X, A), A), X\right) + \gamma L_{Adv}\left(h(f(X, A)), A\right) \tag{4}$$

One of the main contributions of LAFTR is the objective the authors suggest for the adversary. Instead of having an adversary trying to minimize a classification loss by treating the sensitive attributes as labels (which they also consider for comparison purposes), they suggest to base the objective on known metrics from the fairness literature applied to the classifier g. They consider three group fairness metrics: demographic parity, equalized odds and equal opportunity. For each of these metrics, the authors derive an adversarial objective whose value maximized by the adversary upper-bounds the metric evaluated on g, meaning that the lower the adversarial objective is, the more fair the classifier g will be with respect to the considered metric. The idea for deriving the upper bounds is the same for each of the metrics. They first suggest an objective function for the adversary and in each case they show there exists an arbitrary adversary for which the proposed objective is equal to the metric evaluated on g. Since the adversary will be trained to maximize this objective, it follows that the value of its objective will be at least as large as the one of the arbitrary adversary and thus it will upper-bound the metric evaluated on g.

An important point of their approach is the fact that the adversary takes as input the learned representations Z instead of $g(Z)$ because it results in a stronger adversary, i.e. a potentially greater upper bound, allowing to obtain a more fair classifier g. In fact, if the adversary were to work with $g(Z)$ as an input, the authors show that adding the adversarial term would be the same as simply having a regularizing term enforcing the considered metric on the classifier g.

The authors demonstrate how their framework can also be used in the context of learning fair and transferable representations, where the goal is still to have a fair encoder but with the additional constraint that the learned representations should be able to generalize to new unseen labels. This means that a classifier trained on top of these representations for a different classification task than the one considered during the training of the encoder should still have high performance. It should be noted that in this set up, the reconstruction term of the objective is quite important as it allows to preserve more information from the input samples in the learned representations, thus yielding better transferability.

In [2], the end goal is again to obtain a feature extractor f that allows a classifier g trained on top of it to reach high performance and learns features independent of a sensitive attribute / spurious feature to which they refer as the bias b, but for consistency with the rest of the report we will keep the letter A to represent it. The resulting architecture, depicted in Fig. 2, is quite similar to the one from LAFTR if we ignore the decoder module but their starting point is different. In order to obtain representations

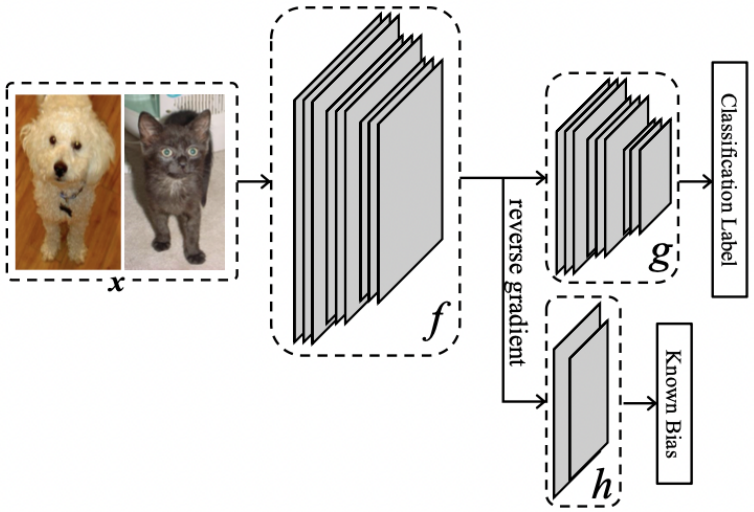

Figure 2: LNTL model.

independent of the sensitive attribute, they suggest minimizing the mutual information between the representations and the sensitive attribute by adding it to the objective function of the network (feature extractor + classifier) as a regularization term, giving the following optimisation problem, where I stands for the mutual information operator, weighted by λ, θ_f and θ_g respectively correspond to the parameters of f and g, and L_c denotes a classification loss such as cross-entropy loss:

$$\min_{\theta_f,\theta_g} \mathbb{E}_{P(X)} \left[L_c(g(f(X)), Y)\right] + \lambda I\left(A(X), f(X)\right) \tag{5}$$

Minimizing the mutual information $I\left(A(X), f(X)\right)$ in this case is equivalent to minimizing the negative conditional entropy $-H(A(X)|f(X))$. However, this requires access to the posterior distribution $P(A(X)|f(X))$ which is unknown, so they propose to learn an approximation $Q(A(X)|f(X))$ to this distribution by means of an additional network, the bias prediction network h, thus reformulating the mutual information term $I(A(X), f(X))$ from Eq. 5 as:

$$L_{MI} = \mathbb{E}_{P(X)} \left[\mathbb{E}_{Q(A(X)|f(X))} \left[log\ Q(A(X)|f(X))\right]\right] \tag{6}$$

The bias prediction network h learns to approximate $P(A(X)|f(X))$ by minimizing the cross-entropy loss between the bias $A(X)$ and its output $h(f(X))$, thus the adversary tries to minimize the following quantity:

$$L_B(\theta_f, \theta_h) = \mathbb{E}_{P(X)} \left[L_c(h(f(X)), A(X))\right] \tag{7}$$

The final objective is then composed of a classification term, the mutual information regularizer and the bias prediction term:

$$\min_{\theta_f,\theta_g} \max_{\theta_h} \mathbb{E}_{P(X)} \left[L_c(g(f(X)), Y) + \lambda \mathbb{E}_{Q(A(X)|f(X))} \left[log\ Q(A(X)|f(X))\right]\right] - \mu L_B(\theta_f, \theta_h) \tag{8}$$

It is important to notice that the feature extractor also maximizes $L_B(\theta_f, \theta_h)$, which in practice is achieved with a gradient reversal layer as shown in their model in Fig. 2.

3.2 Blind Fairness Without Demographics

For the case where subgroup information is not available in the original data, we consider two other papers [3] and [4] respectively authored by Preethi Lahoti et al. and Elliot Creager et al. In this case,

the two implementations are quite different from each other. Indeed, [3] proposes a way of guaranteeing good performance for all subgroups without knowing them at any point of the implementation. However, the approach in [4] consists first in obtaining the subgroup labels of the samples, and then using them in the context of Invariant Risk Minimization (IRM) [5], which is another method seeking to ensure fairness during training by explicitly using the subgroup labels, similarly to BF.

We start with [3], to which we will also refer as ARL for adversarially reweighted learning. The goal here is to achieve high classification performance on every subgroup by focusing on high-loss samples. As the sensitive attribute / spurious feature A of the training data is not known, the authors start by assuming that the sensitive attribute, which can take K values, is correlated with the observed features X and their labels Y, which is usually true. Their idea is then to have a model, which they refer to as the learner h, trained to minimize a weighted classification loss. The learner is trained with an adversary that tries to maximize the classification loss by giving large weights to samples with high loss (a.k.a. the high-risk group), so that the learner can focus on improving its performance on these samples. The resulting optimisation problem is shown below, where θ and ϕ are respectively the parameters of the learner and the adversary, $\lambda_\phi(x_i, y_i)$ is the weight given to sample x_i and l_{ce} denotes the cross-entropy loss:

$$\min_{\theta} \max_{\phi} \sum_{i=1}^{N_{tr}} \lambda_\phi(x_i, y_i) l_{ce}(h_\theta(x_i), y_i) \tag{9}$$

The architecture they present in their paper is shown in Fig. 3. As the adversary's output for each sample is in $(0, 1)$, scaling these values is necessary to use them as weights, mainly to avoid exploding gradients, but also centering them (adding 1) so that even the samples not up-weighted by the adversary contribute to the loss, giving the final weight λ_i for each sample x_i as shown in Fig. 3.

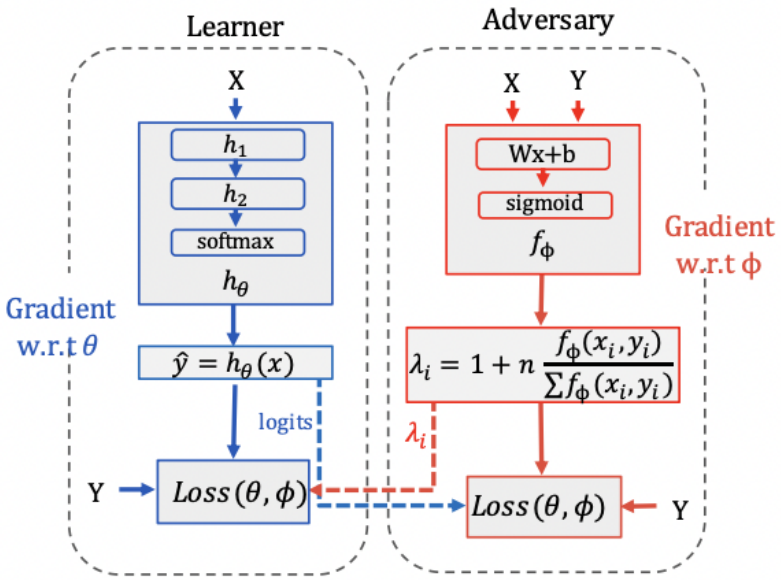

Figure 3: ARL model.

An important point of their approach is the fact that their model is robust to outliers (which could be the result of label noise), so the model will not try to specially focus on them as other approaches can erroneously do (e.g. Distributionally Robust Optimisation (DRO)). This is because by definition, an outlier is a sample with a rarely-occurring combination of X and Y. Since the adversary uses X and Y as input, for it to focus on the outliers would basically mean to output a large value on a single, isolated outlier while outputting small values for all the other samples. Even if this could happen, in cases where there are in fact other groups of samples with high loss, the adversary will probably find that focusing on those samples rather than the outlier allows it to further maximize the loss.

In [4], or environment inference for invariant learning (EIIL) as the title of the paper, the authors

base their work in the paper *Invariant Risk Minimization* (IRM) by Martin Arjovsky et al. [5], which proposes a method for learning representations independent of spurious features. In IRM, the subgroups are called environments and their definition is different from the one of subgroups in this report, so for clarity we will keep the name environment. Environments differ between each other by the degree of correlation between the spurious feature and the class label, rather than by the value of the spurious feature as subgroups do. The goal in IRM is to obtain representations invariant to environments, which they restate as obtaining a representation such that the optimal classifier trained on top of it is *the same across every environment*. In other words, an invariant representation $\Phi(X)$, i.e. independent of a spurious attribute e taking values in $\mathcal{E}_{obs}$, should satisfy the following equality, called the *Environment Invariance Constraint* (EIC), where $\mathcal{H}$ denotes the representation space and Y represents the ground truth labels:

$$\mathbb{E}[Y|\Phi(X) = h, e_1] = \mathbb{E}[Y|\Phi(X) = h, e_2] \ \forall h \in \mathcal{H}, \ \forall e_1, e_2 \in \mathcal{E}_{obs} \tag{10}$$

Similarly to the BF setting, they achieve this goal by making explicit use of environment labels during the training procedure as they add to the classification loss a penalty term, evaluated separately on each environment, equivalent to enforcing EIC. The penalty term becomes 0 for all environments when the same "dummy" classifier (i.e. a classifier outputting the same value as its input) is optimal on all environments. The optimisation problem they are trying to solve is the following one:

$$\min_{\Phi: X \to Y} \sum_{e \in \mathcal{E}_{tr}} R^e(\Phi) + \lambda \left|\left|\nabla_{\bar{w}|\bar{w}=1.0} R^e(\bar{w} \circ \Phi)\right|\right|^2 \tag{11}$$

In this optimisation problem, $R^e(\Phi) = \mathbb{E}_{X^e, Y^e}\left[l(\Phi(X^e), Y^e)\right]$ is the risk/classification loss on environment e [1] and the second term is the penalty equivalent to enforcing EIC with its associated weight λ.

Instead of finding an implementation agnostic of the environment labels as in ARL, the authors of EIIL suggest a method to automatically recover binary environment labels to later use them with models implementing IRM. Their contribution relies on the penalty term introduced in IRM, together with a reference classifier. The reference classifier should be biased with respect to the spurious feature, i.e. it classifies samples based on the spurious features rather than on meaningful/causal features that truly explain the class labels. It can be obtained either by handcrafting a deterministic classifier if prior knowledge about the spurious feature is known, or by training a model with Empirical Risk Minimization (ERM), since training without any advanced techniques on a highly skewed population will also frequently yield a biased classifier. Once the reference classifier is trained, it is fixed, and contributes to the implementation only through the label predictions it produces for the data samples. To infer the environments, they first associate a random probability $q(e|x_i, y_i)$ to each sample (these probabilities are free variables, modelled by a vector $\mathbf{q}$ with a coefficient for each data sample), indicating a soft assignment of sample x_i to environment e, with $e \in \{e_1, e_2\}$ and $q(e_1|x_i, y_i) = 1 - q(e_2|x_i, y_i)$. These probabilities are then optimized to maximally violate EIC by maximizing a relaxed version of the penalty term from IRM where the soft environment assignments $\mathbf{q}$ are used to identify samples of each environment and compute the penalty only on those samples.
To summarize, the environment inference phase consists in:

1. Obtaining the biased predictions $\Phi(X)$ [1] of a reference classifier.
2. Inferring environment labels $\mathbf{q}^* = \underset{\mathbf{q}}{argmax} \left|\left|\nabla_{\bar{w}} \left[\frac{1}{\sum_i q_i(e)} \sum_i q_i(e) l(\bar{w} \circ \Phi(x_i), y_i)\right]\right|\right|$, where l is a classification loss.

As the environments are inferred to maximally violate EIC, the resulting environments correspond to samples where the spurious correlation is strong and samples where it isn't. Indeed, a straight forward way of maximally violating EIC is to take samples yielding the same representation/prediction by the reference classifier (meaning that they most probably share the same spurious feature) and splitting them according to their class labels. For simplicity, let's assume that the reference classifier is very biased so it outputs mainly two representations h_0, h_1 respectively corresponding to samples having the spurious feature $A = 0$ and $A = 1$, and that the spurious feature directly correlates with the class label, i.e. $Y = 0$ samples have mostly $A = 0$ and $Y = 1$ samples have mostly $A = 1$. To maximally violate EIC for samples mapped to h_0, we can put samples with $Y = 0$ (arbitrarily) in environment 0 and samples

[1] Here since a dummy classifier $\bar{w} = 1.0$ is used on top of the representation $\Phi(X)$, the latter can also be interpreted as directly corresponding to label predictions

with $Y = 1$ in environment 1. To maximally violate EIC for samples mapped to h_1, we can put samples with $Y = 1$ in environment 0 and samples with $Y = 0$ in environment 1. Thus, environment 0 contains samples with $Y = 0, A = 1$ or $Y = 1, A = 0$, i.e. samples where the spurious feature weakly correlates with the label, and environment 1 contains samples with $Y = 0, A = 0$ or $Y = 1, A = 1$, i.e. samples where the spurious feature strongly correlates with the label. One might argue that splitting the samples mapped to h_1 in the opposite way, i.e. samples with $Y = 0$ in environment 0 and samples with $Y = 1$ in environment 1 would also maximally violate EIC for h_1 without affecting EIC for h_0, so the inferred environments would then correspond to class labels. For a possible explanation of why this does not happen, it is a good idea to come back to the actual penalty term being maximized. With environments defined by the class labels, this penalty measures the optimality of the dummy classifier on each class for the classification task. Given that we assumed that in each class the majority of samples show a spurious correlation, the dummy classifier will actually have a good average performance since it will only make mistakes on the minorities in each class. In contrast, with environments defined by the strength of the spurious correlation, the dummy classifier will be optimal only on the the environment where the spurious correlation is strong while it will be far from optimal on the the environment where the spurious correlation is weak.

4 Project Goal and Implementation

Similarly to EIIL, the goal of this project is to obtain subgroup labels for a given dataset where they are missing so that the dataset together with the found subgroup labels can be used with models in the BF framework. To this end, we will also make use of a reference classifier giving biased label predictions, so the first step towards the desired goal is to get a biased reference classifier by training a model with ERM. The dataset should contain highly skewed subgroups in each class to easily bias the reference classifier during its training. After training, the reference classifier is fixed and only used to produce a biased class prediction for each sample. The core idea is then to have a model, that we call the subgroup classifier[2], find a partitioning of the data (i.e. the subgroups) such that the reference classifier is most unfair when evaluated with a group-fairness metric given the inferred subgroups.

4.1 Objective Function

We first need to formulate an objective function that the subgroup classifier can optimize and whose extreme value translates into the reference classifier being most unfair. We can transform the equalities that need to be satisfied to achieve demographic parity and equalized odds to gaps that we name demographic parity gap (DPG) and equalized odds gap (EOG) as follows:

$$\text{DPG: } \sum_{y\in\mathcal{Y}} \sum_{\substack{a_k,a_l\in\mathcal{A}\\ a_k\neq a_l}} \left(\frac{\sum_i \mathbb{1}[\hat{y}_i = y] P(\hat{A} = a_k|x_i)}{\sum_i P(\hat{A} = a_k|x_i)} - \frac{\sum_i \mathbb{1}[\hat{y}_i = y] P(\hat{A} = a_l|x_i)}{\sum_i P(\hat{A} = a_l|x_i)} \right)^2 \tag{12}$$

$$\text{EOG: } \sum_{y\in\mathcal{Y}} \sum_{\substack{a_k,a_l\in\mathcal{A}\\ a_k\neq a_l}} \left(\frac{\sum_i \mathbb{1}[\hat{y}_i \neq y]\mathbb{1}[y_i = y] P(\hat{A} = a_k|x_i)}{\sum_i \mathbb{1}[y_i = y] P(\hat{A} = a_k|x_i)} - \frac{\sum_i \mathbb{1}[\hat{y}_i \neq y]\mathbb{1}[y_i = y] P(\hat{A} = a_l|x_i)}{\sum_i \mathbb{1}[y_i = y] P(\hat{A} = a_l|x_i)} \right)^2 \tag{13}$$

To avoid any confusion, here $\hat{y}_i$ is the class label predicted by the reference classifier for sample x_i and $P(\hat{A}|x_i)$ is the output of the subgroup classifier. The sums in each objective can be normalized by the number of gaps considered, which at most is $n * \binom{m}{2}$, recalling that there are n classes and m subgroups. To find a partitioning of the data for the reference classifier to be most unfair with respect to one of these two metrics, the subgroup classifier can maximize the corresponding fairness objective.

4.2 Mutual Information Regularizer

The objectives in Eq. (12) and Eq. (13) are ill-defined in the sense that there is a solution not corresponding to the desired subgroups yielding a higher objective value. This solution corresponds to the subgroup classifier mimicking the reference classifier, i.e. $\hat{A} = \hat{Y}$. Indeed, for demographic parity, if

[2]The term subgroup classifier will always include the feature extraction and the classification layers of the model unless otherwise specified.

$\hat{A} = \hat{Y}$, $P(\hat{Y} = y|\hat{A})$ will be 1 when $\hat{A} = y$ and 0 otherwise. A similar argument holds for equalized odds. To mitigate this issue without completely redefining the objectives, we add to the objective a regularizer so that the mutual information between the learnt subgroups and the predictions of the reference classifier is minimized, and refer to the LNTL paper for its practical implementation. Therefore, in addition to maximizing the chosen fairness objective, we want the subgroup classifier to minimize the following quantity, where I and H respectively correspond to the mutual information and the entropy operators.

$$I(\hat{Y}(X); \hat{A}(X)) = H(\hat{Y}(X)) - H(\hat{Y}(X)|\hat{A}(X)) \tag{14}$$

$H(\hat{Y}(X))$ can be left out of the optimization procedure as it does not depend on the subgroup classifier parameters, and we thus focus on the negative conditional entropy:

$$-H(\hat{Y}|\hat{A}) = \sum_a P(\hat{A} = a) \sum_y P(\hat{Y} = y|\hat{A} = a) log\left(P(\hat{Y} = y|\hat{A} = a)\right) \tag{15}$$

Assuming that $P(\hat{A})$ is uniform, which is not necessarily the case but it will be for the experiments conducted, we can add to the implementation an auxiliary network that learns $P(\hat{Y} = y|\hat{A} = a)$ so that the subgroup classifier can directly minimize Eq. (15). The auxiliary network will learn $P(\hat{Y} = y|\hat{A} = a)$ by minimizing the cross-entropy loss between the predictions of the reference classifier (considered to be the ground truth in this case) and its own predictions.

4.3 Final Architecture

Following the implementation of [2], we jointly train the subgroup classifier and the auxiliary network in an adversarial way. The resulting architecture is depicted in Fig. 4, which attempts to solve the following optimisation problem, where $DPG(\hat{Y}, f(g(X)))$ can be replaced by $EOG(\hat{Y}, Y, f(g(X)))$:

$$\min_{f,g} \max_{h} \; -DPG(\hat{Y}, f(g(X))) - \lambda H(\hat{Y}|\hat{A}) - L_{ce}(\hat{Y}, h(f(X))) \tag{16}$$

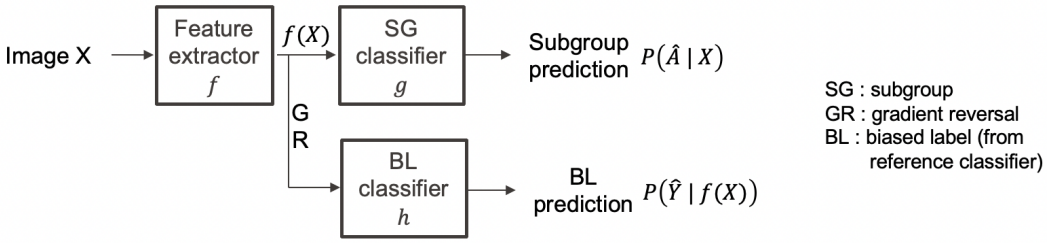

Figure 4: Final architecture for subgroup prediction.

In practice, each batch of samples first goes through the feature extraction network f to obtain its representation $f(X)$. The latter is then fed to the classification layers g of the subgroup classifier to find the partitioning of the data making the reference classifier most unfair. The representation $f(X)$ also goes through the auxiliary network h to obtain $P(\hat{Y}|f(X))$ to be able to compute the mutual information regularizer. It should be noted that in the developments for the latter in the previous subsection, we were using $P(\hat{Y}|\hat{A})$, but now we are working with $P(\hat{Y}|f(X))$. The intuition for this is that g won't find the problematic solution $\hat{A} = \hat{Y}$ if the feature representation $f(X)$ is independent of $\hat{Y}$. f and g then take a gradient step to minimize the negated fairness objective and the mutual information regularizer. Secondly, the batch of samples goes again through f to obtain $f(X)$. The gradient reversal technique is then applied to $f(X)$ before feeding it through h. The latter will try to predict the biased predictions of the reference classifier by taking a gradient step to minimize the cross-entropy loss. In addition, the use of the gradient reversal technique means that the feature extractor f will also take a gradient step to maximize the cross-entropy loss, to make it harder for h to make its predictions.

5 Experiments

In this section we review the experiments that were conducted to test the implementation and their results.

5.1 Datasets

As the approach is quite experimental, the MNIST dataset was chosen since it has extensively been used in ML for different techniques and is known for its simplicity, together with its slightly more complex drop-in replacement FashionMNIST. To introduce bias in the data, the images were colored so that the color acts as the spurious feature, i.e. in the train set different class labels are highly correlated with different colors, and the subgroups we are trying to recover are sets of samples sharing the same color regardless of their class label. We refer to the biased versions of the datasets as CMNIST for Colored MNIST and CFMNIST for Colored FashionMNIST. For the test sets, the spurious correlations for each class are reversed.

The first and simplest version of the modified datasets was realised by keeping two classes (4/9 digits for MNIST, trouser/pullover for FashionMNIST) from the original datasets and introducing two colors, yielding a reduced version of the datasets with statistical bias as shown in Table 1[3].

	Color0	Color1
Class0	0.95	0.05
Class1	0.05	0.95

Table 1: Subgroup proportions per class (2 classes, 2 subgroups).

To make the task a bit harder, a similar dataset was constructed this time with 3 classes (4/7/9 for MNIST, t-shirt/pullover/shirt for FashionMNIST) and 3 colors, as shown in Table 2[3].

	Color0	Color1	Color2
Class0	0.90	0.05	0.05
Class1	0.05	0.90	0.05
Class2	0.05	0.05	0.90

Table 2: Subgroup proportions per class (3 classes, 3 subgroups).

Finally, a third version with all the classes and only two subgroups was constructed. In this case, half of the classes are represented in majority by color0 while the other half is represented in majority by color1, as shown in Table 3[3].

	Color0	Color1
Class0...4	0.95	0.05
Class5...9	0.05	0.95

Table 3: Subgroup proportions per class (10 classes, 2 subgroups).

5.2 Models

All the models used (f, g and h) were implemented with fully connected layers and optimized with stochastic gradient descent. In all experiments, the learning rate for the subgroup classifier (f and g) is 0.01, the one of the BL classifier h is 0.001, the weight given to the mutual information regularizer is 1.0, f has two layers and g and h have a single one. The sizes of the fully connected layers, the batch sizes and the number of training epochs are specified in Table 4 for each version of the two datasets considered.

5.3 Results

All the results presented in what follows (accuracies, learning curves) have been averaged over 10 runs.

[3] In the table, the indices used for the classes have been renamed to range between 0 and the number of considered classes.

dataset version	CMNIST	CFMNIST
2 classes, 2 subgroups	f: 588x64, 64x64 g: 64x2 h: 64x2 30 epochs batch size 4096	f: 588x64, 64x64 g: 64x2 h: 64x2 25 epochs batch size 4096
3 classes, 3 subgroups	f: 588x64, 64x64 g: 64x3 h: 64x3 30 epochs batch size 512	f: 588x64, 64x64 g: 64x3 h: 64x3 30 epochs batch size 512
10 classes, 2 subgroups	f: 588x64, 64x64 g: 64x2 h: 64x10 10 epochs batch size 512	f: 588x128, 128x128 g: 128x2 h: 128x10 5 epochs batch size 512

Table 4: Models configuration.

5.3.1 2 classes, 2 subgroups

The first tests were done with 2 classes and 2 subgroups, the simplest version of CMNIST/CFMNIST. Table 5 shows the average accuracies and their standard deviations obtained on the train and test sets for CMNIST and CFMNIST, using the demographic parity gap.

Dataset	Train accuracy	Test accuracy
CMNIST	1.0 ± 0.0	1.0 ± 0.0
CFMNIST	0.987 ± 0.012	0.980 ± 0.031

Table 5: Average performance (2 classes, 2 subgroups).

Overall the accuracies obtained are quite good in particular for CMNIST where they are perfect on both the train and test sets. However, this is not entirely surprising as this version of the datasets is quite simple and the reference classifier is easily biased. For instance, the target demographic parity gap (i.e. the value of the demographic parity gap when evaluated with the true subgroups) is near 0.999 for CMNIST whereas it is 1.0 when taking as subgroups the predictions made by the reference classifier.

The learning curves are shown in Fig. 5. With both datasets, f and g manage to properly maximize the demographic parity gap and stay near the target without reaching 1.0, i.e. without learning the problematic solution $\hat{Y} = \hat{A}$. As the demographic parity increases, the mutual information regularizer first increases, which is expected, and then f and g try to reduce it while keeping the demographic parity gap high. Similarly, the cross-entropy loss for h increases during the first epochs. As the demographic parity gap becomes closer to its target, h starts to be able to improve its classification performance. Since in this case, the problematic solution and the desired one are very close, it is also not surprising that by the end of the training the loss for h is not very high. In terms of accuracy, at the end of the training the adversary would sometimes show a high accuracy but other times the accuracy would be much closer to 0.5, i.e. random guessing, which is more desirable. Increasing the number of epochs helped bringing the accuracy of h to values closer to 0.5 but a configuration where this would happen systematically was not found.

5.3.2 3 classes, 3 subgroups

Next, we study if the implementation also works for the case of non-binary classes and subgroups by experimenting on the version of the datasets with 3 classes and 3 subgroups. Table 6 shows the average accuracies and their standard deviations obtained on the train and test sets for CMNIST and CFMNIST, using the demographic parity gap.

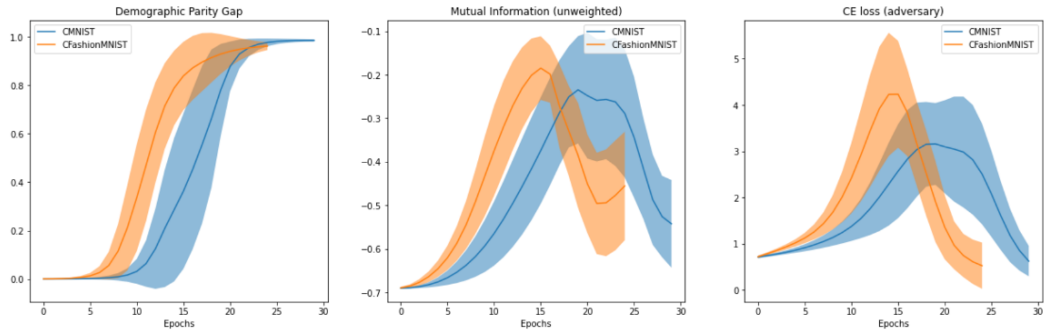

Figure 5: Learning curves (2 classes, 2 subgroups).

Dataset	Train accuracy	Test accuracy
CMNIST	0.857 ± 0.287	0.813 ± 0.371
CFMNIST	0.899 ± 0.298	0.899 ± 0.298

Table 6: Average performance (3 classes, 3 subgroups).

With 3 classes and 3 subgroups, the performance is not as good as in the binary case. The problem here is the stability of the trainging procedure: as shown by the standard deviations, the performance could vary considerably. In general, over 10 runs, one or two runs would fail completely [4] while the rest of the runs would reach train and test accuracies ≥ 0.99. With CMNIST for instance, the only way that was found to get constant results in this experiment was by using a batch size equal to the whole train set and by fixing the initialization of the weights of the models.

Nevertheless, the learning curves shown in Fig. 6 are more interesting than in the previous experiment. Here the behaviour of whole implementation is closer to the desired one for both datasets: initially the demographic parity gap increases (potentially beyond the target value) together with the mutual information and both peak around 5 epochs. At that point, the loss for h is at its lowest. After the 5th epoch, the mutual information starts to decrease again while the cross-entropy loss for h increases a bit. This part corresponds to f and g refining themselves in order to maintain a high demographic parity gap, a low mutual information but at the same time making it harder for h to continue minimizing its loss. However, the demographic parity gap does decrease during the beginning of the refining phase until it is stabilized.

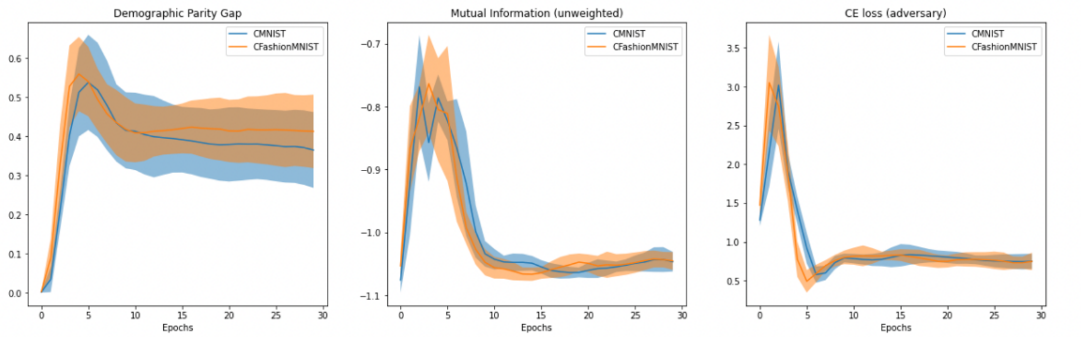

Figure 6: Learning curves (3 classes, 3 subgroups).

The target demographic parity gap for CMNIST in this experiment was 0.6374. The average value in the last epochs is a bit far from the target (even if we ignore the failed runs which have a much lower value), as shown in the corresponding plot, despite being able to reach high accuracies on the train and test sets on the successful runs. Although this might first appear inconsistent, the reason for this is that

[4] When this happens, the subgroup classifier does not learn anything and maps every sample to the same value.

during the refining step mentioned before, the subgroup classifier can become less confident in its output for certain samples, which in turn will decrease the demographic parity gap as those samples will count less. However, since the final prediction is based on a threshold, if the confidence level doesn't change enough to go over/below the threshold considered, the final performance can still be high.

5.3.3 10 classes, 2 subgroups

The last experiment considered with CMNIST and CFMNIST is the one with all the classes and only 2 subgroups. The average accuracies are shown in Table 7. Here, stable results were obtained again for both datasets. However, the performance is quite different. With CMNIST, the implementation works really well similarly to the first experiment, whereas with CFMNIST the train accuracy remains relatively low and in comparison the test accuracy is surprisingly high. The main explanation for this resides in the reference classifier. Indeed, with CMNIST, it was relatively easy to bias the reference classifier so that it relies a lot on the color to make its prediction, making it very accurate on the subgroups whose spurious feature is highly correlated with the class labels. With CFMNIST, however, this was harder to achieve, there were some classes where the reference classifier had good performance on both subgroups, i.e. it was less biased.

Dataset	Train accuracy	Test accuracy
CMNIST	0.999 ± 8e-05	1.0 ± 0.0
CFMNIST	0.720 ± 0.097	0.902 ± 0.029

Table 7: Average performance (10 classes, 2 subgroups).

A good reason to include this experiment, other than all the classes being used, is that in this case only the equalized odds gap was shown to work. Indeed, in the two previous experiments, EOG could be used instead of DPG to obtain very similar results, although they could be less reliable (i.e. the implementation could fail more often). Here, because DPG conditions on the subgroup and the fact that we had more than one class mainly represented by the same subgroup, this objective was less expressive, and the subgroup classifier would have a hard time learning the subgroups. However, with the same configuration but replacing DPG with EOG, the implementation reached high performance for CMNIST and an acceptable one for CFMNIST considering the reference classifier used.

The learning curves, depicted in Fig. 7 are also very close to the first experiment, with a minor drop in EOG after the first epochs, the mutual information increasing first because EOG increases too and then decreasing again, and similarly for the loss of h.

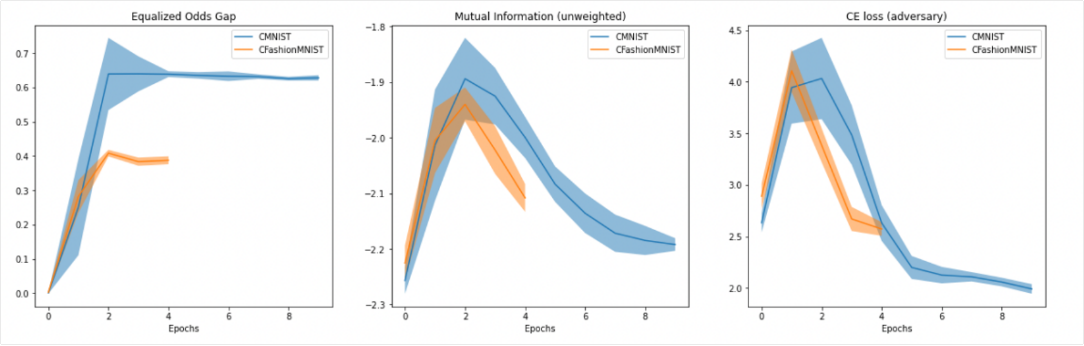

Figure 7: Learning curves (10 classes, 2 subgroups).

5.4 Additional Experiments

Additional experiments were conducted to test the implementation on more complex datasets and models. The additional datasets considered were CIFAR-10S [6] and CelebA [7]. CIFAR-10S is a modified version of the CIFAR-10 dataset, where gray-scale images have been introduced to obtain a dataset with

a bias similar to the one for the experiments with 10 classes and 2 subgroups with CMNIST/CFMNIST, i.e. in the train set, 5 classes contain mainly gray-scale images whereas the other 5 contain mainly color images. CelebA is a dataset of images of celebrities where several physical traits correlate with other features of the persons represented such as gender. In the version that was used, the class labels were blond/non-blond and the subgroups were males/females, with females being mostly blond and males mostly non-blond in the train set. For this two datasets, g and h were still implemented with fully connected layers. However, the feature extractor f was modelled by a convolutional neural network (CNN).

Experiments were first done with a reduced version of CIFAR-10S where only two classes where kept (deers and horses), to have experimental conditions similar to the first experiment described above. As with the second experiment above, the main problem was the training instability, which could mainly come from the adversarial training framework as training a subgroup classifier to only maximize DPG yielded more consistent results, with 1 run failing over 10. However, the performance without the adversary (i.e. without the mutual information regularizer) remained quite poor when compared to the one in the first experiment above, reaching on average 0.8 accuracy on the train set and 0.7 on the test set. 10-fold cross-validation was performed with the mutual information regularizer in order to find the combination of hyper-parameters (learning rates, batch sizes, mutual information regularizer weight, number of epochs) that would produce the most stable results. The most stable configuration worked on only 3 out of the 10 folds and had a learning rate of 0.1 for f and g and of 0.001 for h, a batch size of 1500 samples, a mutual information weight of 1.0 and 90 epochs.

As none of the experiments with the reduced version of CIFAR-10S were successful, to check whether it could be linked to the dataset itself, CelebA was considered. Very few tests where run on this dataset, but the same instability during training was observed, suggesting that as expected it had nothing to do with the dataset but rather with the implementation itself.

6 Conclusion

To conclude, the proposed implementation to obtain subgroup labels for a dataset containing different subgroups has been shown to work on different experiments with two datasets, CMNIST and CFMNIST. The most successful experiment in terms of performance and stability was also the simplest one, with only 2 classes and 2 subgroups, closely followed by the third experiment with all the classes and 2 subgroups. The second experiment, with 3 classes and 3 colors was the worst in terms of stability. These experiments confirm that the suggested implementation can be used for the task of obtaining subgroup labels for the data samples, but also exposed its main flaw, i.e. the stability of the training procedure. However, this shouldn't rule out completely the validity of the proposed method as further investigation could lead to finding a truly stable implementation. In particular, more advanced techniques such as adaptive learning rates and regularizer weights, norm penalties or batch normalisation could be key in stabilizing the training.
A major drawback of this work is that it relies on a reference classifier that should be biased with respect to the spurious feature. The method works better the more the reference classifier is biased, giving the reference classifier an important role to play, probably too important for the cases where very little prior information about the spurious feature is known, i.e. when it will be difficult to judge if the reference classifier is indeed biased.
Despite achieving good results on some of the experiments, one can't help but wonder if most of the flaws could be attenuated simply by properly defining an objective whose optimal value coincides with the target value, rather than trying to get to the target by means of an additional quantity to optimize. This alternative objective could also make use for instance of an individual-fairness metric rather than a group-fairness metric to get a finer-grain loss.
In addition, other datasets closer to real-world datasets could be used to further challenge the implementation. For instance, datasets where not all the subgroups are present in each class, where the minority subgroups are present at different rates in different classes or where there exists more than one binary spurious feature instead of a single non-binary one. Of course, getting the implementation working on CIFAR-10S and CelebA with CNNs for the feature extractor would be of most importance. Lastly, recalling that in the development of the mutual information regularizer, the distribution of the spurious feature was assumed to be uniform, an interesting next step could also be to generalize this regularizer for non-uniform subgroups.

References

[1] David Madras et al., Learning Adversarially Fair and Transferable Representations. *arXiv:1802.06309v3*, 2018.

[2] Byungju Kim et al., Learning Not to Learn: Training Deep Neural Networks with Biased Data. *arXiv:1812.10352v2*, 2019.

[3] Preethi Lahoti et al., Fairness without Demographics through Adversarially Reweighted Learning. *arXiv:2006.13114v3*, 2020.

[4] Elliot Creager et al., Environment Inference for Invariant Learning. *arXiv:2010.07249v5*, 2021.

[5] Martin Arjovsky et al., Invariant Risk Minimization. *arXiv:1907.02893v3*, 2020.

[6] Zeyu Wang et al., Towards Fairness in Visual Recognition: Effective Strategies for Bias Mitigation *arXiv:1911.11834v2*, 2020.

[7] Ziwei Liu et al., Deep learning face attributes in the wild *arXiv:1411.7766v3*, 2015.