FAIRBATCH: BATCH SELECTION FOR MODEL FAIRNESS

Anonymous authors
Paper under double-blind review

ABSTRACT

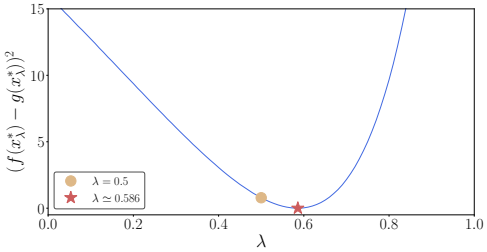
Model fairness is becoming essential in any deep learning application in order to reduce discrimination. While many unfairness mitigation techniques have recently been proposed, they require fairness expertise and drastic changes in either the training data generation or model training algorithm, which limit their applicability especially to existing deep learning pipelines. We contend that such overheads can be reduced and propose FairBatch, which targets mini-batch stochastic gradient descent model training and takes the novel approach of only modifying the batch selection to improve fairness and accuracy. We first formalize fair batch selection as a bilevel optimization problem for fairness and accuracy and show that the objective function satisfies quasi-convexity and can thus be provably solved using normalized gradient descent with multiple model trainings. We then propose an efficient approximate algorithm that runs within a single model training where FairBatch adjusts the sensitive group ratios in each batch based on the fairness of the current intermediate model. We show how FairBatch can be used with the equal opportunity fairness measure and extend it to support two other representative group fairness measures: equalized odds, and demographic parity. FairBatch is not only easy to use (replace a line of code), but is efficient and surprisingly effective. In our experiments, FairBatch shows comparable fairness and accuracy results against state-of-the-art pre-processing and in-processing unfairness mitigation techniques and an order of magnitude faster than similar boosting approaches (e.g., AdaFair) that train multiple models. We also demonstrate that FairBatch can improve the fairness of any non-fair model (e.g., ResNet-18) and be integrated with existing batch selection techniques that use importance sampling for faster convergence.

1 INTRODUCTION

Model fairness is becoming an essential aspect of any deep learning application. Fairness issues occur in sensitive applications like healthcare, finance, and self-driving cars where a trained model must not discriminate against users based on age, gender, or race. It is not just the new applications that need fairness, but also existing and even legacy deep learning applications as well.

While many unfairness mitigation techniques have recently been proposed to improve fairness, a major problem is that they can be very difficult to implement especially for existing deep learning pipelines. There are two popular unfairness mitigation approaches: pre-processing where the training data is debiased through data generation (Choi et al., 2020) or re-weighting (Jiang & Nachum, 2020) and in-processing where the model training algorithm is improved using fairness objectives (Zafar et al., 2017a), adversarial training (Zhang et al., 2018), or boosting (Iosifidis & Ntoutsi, 2019). In any of these approaches, a major part of the deep learning pipeline needs to be re-engineered, which may be infeasible in practice especially for users without fairness expertise.

We contend that such overheads can be reduced and propose FairBatch, which targets the common scenario of mini-batch stochastic gradient descent (SGD) model training and takes the novel approach of only changing the batch selection part to improve the model's fairness and accuracy. It is well known that careful batch selection using importance sampling can lead to faster convergence of model training (Loshchilov & Hutter, 2016; Alain et al., 2016; Stich et al., 2017; Csiba & Richtárik, 2018; Katharopoulos & Fleuret, 2018; Johnson & Guestrin, 2018). Our novel contribution is a batch selection technique to improve model fairness as well.



(a) FairBatch finding the best λ [[Will be change to FairBatch with a grid search result]]

```
fairloader = FairBatch (model, criterion, train_data,
                        batch_size, alpha, target_fairness)

for epoch in range(epochs):
  for i, data in enumerate(fairloader):
    # get the inputs; data is a list of [inputs, labels]
    inputs, labels = data

    ... model forward, backward, and optimization ...
```

(b) PyTorch sample code using FairBatch.

Figure 1: The left figure illustrates how FairBatch quickly optimizes fairness by solving a quasi-convex optimization problem. The right figure shows how FairBatch can easily be used by a PyTorch deep learning pipeline with a single-line code change (blue color).

We first formalize the problem of fair batch selection as a bilevel optimization problem for maximizing fairness and accuracy together. We prove that this problem is quasi-convex, which leads us to a normalized gradient descent algorithm that is known to exactly solve the problem. This algorithm requires multiple model trainings, so we then propose an approximate algorithm that effectively fits the gradient descent steps within a single model training. The key idea is to adjust the portions of sensitive groups within each batch, based on the fairness of the current intermediate model.

FairBatch naturally supports equal opportunity (Hardt et al., 2016) fairness. For example, suppose we would like to predict whether male and female criminals re-offend in the future. Equal opportunity means that the model makes equally-accurate predictions for criminals that have positive labels (i.e., actually re-offend). During model training, if the accuracy for male is currently worse than female, FairBatch gradually increases the ratio of male examples in the next batches (see Section 3 for details). Figure 1a shows how FairBatch quickly adjusts the sensitive group ratio of a batch after a few epochs in one of our experiments. In addition, we extend FairBatch to support two other popular group fairness measures: demographic parity (Feldman et al., 2015) and equalized odds (Hardt et al., 2016).

In our experiments, we show that FairBatch has a surprisingly comparable performance to state-of-the-art pre-processing and in-processing unfairness mitigation techniques. Also, FairBatch is an order of magnitude faster than similar boosting and re-weighting techniques (e.g., AdaFair (Iosifidis & Ntoutsi, 2019)) that require multiple model trainings.

We also emphasize the practical aspect of FairBatch. First, FairBatch is easy to use where it can be plugged into any deep learning pipeline with minimal code changes. For example, Figure 1b shows PyTorch sample code where just one line of code needs to be changed to use FairBatch. FairBatch can also improve the fairness of any existing non-fair model (e.g., ResNet-18) and be combined with other batch selection techniques that use importance sampling for faster convergence.

In the following sections, we formulate the bilevel optimization problem, propose FairBatch, evaluate it with experiments, and present related work.

2 Bilevel Optimization for Fairness and Accuracy

Preliminaries Suppose a classifier receives an example $x \in X$ and returns a prediction $\hat{y}$.

We consider *group fairness* where the goal is to prevent a model from discriminating between *sensitive groups* (e.g., men versus women). The sensitive groups are determined with a *sensitive attribute* (e.g., gender). For example, equal opportunity (Hardt et al., 2016) states that a model is fair if two sensitive groups of examples with positive labels have similar positive prediction rates, which means they have the same accuracies as well:

$$\Pr\Big(\hat{Y} = \hat{y} | Y = 1, Z = z\Big) = \Pr\Big(\hat{Y} = \hat{y} | Y = 1\Big), \forall \hat{y} \in \hat{\mathcal{Y}}, z \in \mathcal{Z} \tag{1}$$

where Z is the set of sensitive groups, $\hat{Y}$ is the model prediction, and Y is the label. We will cover other group fairness measures in Section 4.

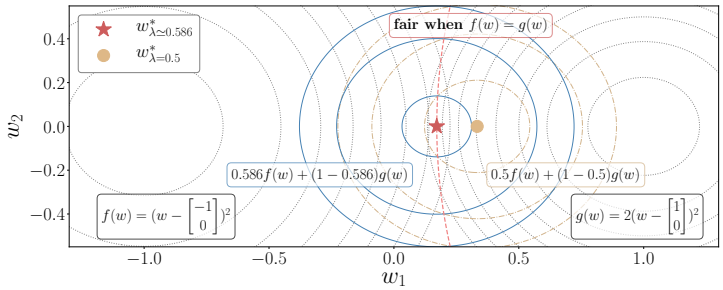


Figure 2: Optimal λ achieving perfect fairness. It shows that the original optimization ($\lambda = 0.5$) gets an unfair solution (beige circle), but it can achieve a fair result by finding optimal λ (red star).

Problem Definition Consider the following example:

$$\min_w f(w) + g(w), \quad \text{where } f(w) = \left\| w - \begin{bmatrix}-1\\0\end{bmatrix} \right\|_2^2, \quad g(w) = 2\left\| w - \begin{bmatrix}1\\0\end{bmatrix} \right\|_2^2.$$

It is easy to see that $w^\star = \begin{bmatrix}1/3\\0\end{bmatrix} \approx \begin{bmatrix}0.333\\0\end{bmatrix}$. Thus, we have $f(w^\star) = 16/9$, $g(w^\star) = 8/9$ and hence $f(w^\star) + g(w^\star) = 24/9 \approx 2.667$. If we view $f(w)$ and $g(w)$ as the loss function for each of two demographic groups and w as the model parameter, one can see that this optimization problem is equivalent to empirical risk minimization (ERM), and its solution $w^\star$ is generally unfair.

One can make this vanilla ERM 'fair' by reweighting the loss terms. For instance, in the previous example, the group associated with $f(\cdot)$ was disproportionally treated compared to the group associated with $g(\cdot)$. To fix this, one can 'over-charge' the cost of errors associated with this underrepresented group by reweighting $f(\cdot)$ with a higher weight than $g(\cdot)$. However, if we assign a too large weight for $f(\cdot)$, the optimal solution will become biased against the group associated with $g(\cdot)$. Therefore, one must find proper weights that strike the critical balance between $f(\cdot)$ and $g(\cdot)$.

Motivated by this observation, we study the following bilevel optimization problem:

$$\min_\lambda F(\lambda) := (f(w^\star_\lambda) - g(w^\star_\lambda))^2, \quad \text{where } w^\star_\lambda := \arg\min_w \lambda f(w) + (1-\lambda)g(w) + h(w).$$

The inner optimization problem minimizes the weighted sum of $f(\cdot)$ and $g(\cdot)$ plus $h(\cdot)$. Here, $f(\cdot)$ and $g($

Note that this is called a bilevel optimization problem because the variable $w^\star_\lambda$, which shows up in the outer optimization problem, is the optimal solution of the inner optimization problem. In addition, equal opportunity naturally fits into this problem where f and g can be the loss functions of the two sensitive groups for $Y = 1$, and h the sum of loss functions of the two sensitive groups for $Y = 0$.

Let us assume for simplicity that $h(w) = 0$. Then the inner problem has a closed-form solution: $w^\star_\lambda = (\frac{2-3\lambda}{2-\lambda}, 0)$. By plugging this expression into the outer optimization, we have $\min_\lambda \frac{64(\lambda^2-4\lambda+2)^2}{(\lambda-2)^4}$. One can easily show that the optimal solution to this problem is $\lambda^* = 2 - \sqrt{2} \approx 0.586$. That is, if we minimize $0.586 \cdot f(w) + (1 - 0.586) \cdot g(w)$ instead of $f(w) + g(w)$, the two terms will have the same value at the optimal value of w, i.e., perfect fairness is achieved at $w^\star$.

More specifically, when $\lambda = 2 - \sqrt{2}$, $w^\star_\lambda = \frac{3\sqrt{2}-4}{\sqrt{2}} \approx 0.172$, so $f(w^\star_\lambda) = g(w^\star_\lambda) \approx 1.373$. The total loss becomes 2.745, which is slightly larger than the original total loss 2.667, and this can be viewed as the cost of achieving perfect fairness. Also, note that the value of $f(w^\star)$ decreased and the value of $g(w^\star)$ increased to have a balance between them.

While this specific example had a closed-form solution, solving a bilevel optimization is difficult in general, i.e., they are usually NP-hard. However, the problem of our interest has a very special structure: the inner problem has a simple weighted sum. In Section 3, we design an algorithm that *provably* solves this class of bilevel optimization problems.

3 FAIRBATCH

Using the quasi-convex property of $F(\lambda)$, we propose an exact algorithm that provably optimizes it and an approximate algorithm that does not necessarily have the theoretical guarantees, but nonetheless performs well in practice.

3.1 EXACT ALGORITHM

One natural attempt to solve the bilevel optimization problem is a gradient-based algorithm. That is, we compute the gradient of the outer objective w.r.t. λ to update the value of λ. However, in order for such an attempt to work, we need to first check whether $F(\lambda)$ is convex or so. Unfortunately, the outer objective function is not convex in general. Even when we assume the inner objective functions $f(\cdot)$ and $g(\cdot)$ are convex (or even strongly convex), the outer function may not be convex. See Appendix

Fortunately, this function still has a good property: It has a unique global optimum, which can be found by (some variants of) gradient-based methods. Indeed, one can show that this function satisfies *quasi-convexity*, i.e., a relaxed version of convexity.

Lemma 1 (Quasi-convexity of $F(\lambda)$ (informal))**.** *Under mild conditions on $f(\cdot)$, $g(\cdot)$ and $h(\cdot)$, the outer objective function $F(\lambda)$ is quasi-convex, and sign $\left(\frac{\mathrm{d}F(\lambda)}{\mathrm{d}\lambda}\right) = \mathbb{1}(f(w_\lambda^*) \leq g(w_\lambda^*))$.*

We defer the proof to the supplementary.

Therefore, we have the following optimization method:

$$\lambda^{(t+1)} = \lambda^{(t)} - \alpha_t \cdot \mathbb{1}(f(w_\lambda^\star) \geq g(w_\lambda^\star)). \tag{2}$$

That is, we iteratively update λ by increasing its value if $f(w_\lambda^\star) < g(w_\lambda^\star)$ and decreasing its value otherwise. This algorithm's convergence guarantee is given in the following lemma, a straightforward application of the famous results of Nesterov (Nesterov, 1984; Hazan et al., 2015).

Lemma 2. *(Informal) Running the sign gradient descent algorithm (equation 2) with T iterations, we have $F(\lambda^{(T)}) - F(\lambda^\star) \leq \frac{1}{\sqrt{T}}$.*

3.2 APPROXIMATION ALGORITHM

While the exact algorithm has a provable convergence guarantee, computing $w_\lambda^\star$ for every iteration is computationally heavy. Instead, one can run a fixed number of gradient steps to approximate $\mathbb{1}(f(w_\lambda^\star) \geq g(w_\lambda^\star))$. That is,

$$\mathbb{1}(f(w_\lambda^\star) \geq g(w_\lambda^\star)) \approx \mathbb{1}(f(w_\lambda^{(t)}) \geq g(w_\lambda^{(t)})),$$

where

$$\begin{aligned}
w_\lambda^{(1)} &= w_\lambda^{(0)} - \gamma_1(f'(w_\lambda^{(0)}) + \lambda g'(w_\lambda^{(0)})),\\
w_\lambda^{(2)} &= w_\lambda^{(1)} - \gamma_2(f'(w_\lambda^{(1)}) + \lambda g'(w_\lambda^{(1)})),\\
&\cdots\\
w_\lambda^{(t)} &= w_\lambda^{(t-1)} - \gamma_{t-1}(f'(w_\lambda^{(t-1)}) + \lambda g'(w_\lambda^{(t-1)})).
\end{aligned}$$

In other words, we alternate the outer optimization problem and the inner optimization problem as follows. One can start with approximately solving the inner optimization problem via SGD or minibatch SGD. After a certain number of iterations (or epochs), one can determine the sign of the outer objective function's gradient. With this sign information, one can update the λ value. This procedure can be repeated until λ converges to $\lambda^\star$.

When minibatch SGD is used, one can simply use a weighted sampling strategy to *simulate* the objective function $\lambda f(w)+(1-\lambda)g(w)$. Assume that $f(w) = \sum_{i=1}^{n_f} f_i(w)$ and $g(w) = \sum_{i=1}^{n_g} g_i(w)$. In order to have an unbiased gradient estimator, one can draw minibatches such that a $\frac{\lambda n_f}{\lambda n_f + (1-\lambda) n_g}$-fraction of the chosen samples comes from f_i's and a $\frac{(1-\lambda) n_g}{\lambda n_f + (1-\lambda) n_g}$-fraction comes from g_i's.

Algorithm 1: Adaptive adjustment of class ratio within batch based on λ w.r.t. equal opportunity.

Input: Intermediate model, criterion, train data (x_{train}, z_{train}, y_{train}),
previous lambda λ_{t-1}, and FairBatch's learning rate α
output = model (x_{train})
$loss$ = criterion (output, y_{train})

$$\lambda_t = \begin{cases} \lambda_{t-1} + \alpha, & \text{if mean}(loss[(y=1,z=0)]) > \text{mean}(loss[(y=1,z=1)]) \\ \lambda_{t-1} - \alpha, & \text{if mean}(loss[(y=1,z=0)]) < \text{mean}(loss[(y=1,z=1)]) \\ \lambda_{t-1}, & \text{otherwise} \end{cases}$$

Output : Next lambda λ_t

This strategy leads us to Algorithm 1 where for each epoch, the ratios of sensitive groups within the batch is adjusted based on the current model losses on the groups. We set α to 0.1 in our experiments. Algorithm 1 only adds a constant overhead of computation on top of model training.

4 Extensions to Other Fairness Measures

We now extend FairBatch to two other popular group fairness measures: demographic parity and equalized odds. While our algorithmic extensions do not exactly fit into the bilevel optimization framework, they use the same principles of adjusting the class ratios per batch and are effective. More details of the algorithms are in the supplementary.

Equalized Odds A model is fair if two sensitive groups with the same true label have similar accuracies, for each true label class:

$$\Pr\Big(\hat{Y} = \hat{y} | Y = y, Z = z\Big) = \Pr\Big(\hat{Y} = \hat{y} | Y = y\Big), \forall \hat{y} \in \hat{\mathcal{Y}}, y \in \mathcal{Y}, z \in \mathcal{Z}. \tag{3}$$

Compared to equal opportunity, we extend Algorithm 1 to adjust the sensitive group ratios for two cases: $Y = 0$ and $Y = 1$ separately.

Demographic Parity A model is fair if two sensitive groups have the same positive prediction rates regardless of the label:

$$\Pr\Big(\hat{Y} = 1 | Z = z\Big) = \Pr\Big(\hat{Y} = 1\Big), \forall z \in \mathcal{Z} \tag{4}$$

To increase the positive prediction rate of a sensitive group z, we extend Algorithm 1 to increase the ratio of ($Y = 1$, $Z = z$) examples to make the model "overfit" and give more positive predictions.

5 Experiments

We provide experimental results for FairBatch. We measure equal opportunity (EO) by taking the difference $\max_{\forall z \in \mathcal{Z}, \hat{y} \in \hat{\mathcal{Y}}} |\Pr\Big(\hat{Y} = \hat{y} | Z = z, Y = 1\Big) - \Pr\Big(\hat{Y} = \hat{y} | Y = 1\Big)|$ where the smaller the difference, the fairer. The smaller the violation, the fairer. The metrics for equalized odds (ED) and demographic parity (DP) are defined analogously.

Datasets We generate a synthetic dataset of 3,000 examples with two non-sensitive attributes x_1 and x_2, a binary sensitive attribute z, and a binary label y, using a method similar to the algorithm proposed by (Zafar et al., 2017a). The (x_1, x_2, y) attributes are generated based on the two Gaussian distributions: $(x_1, x_2)|y = 0 \sim \mathcal{N}([-2; -2], [10, 1; 1, 3])$ and $(x_1, x_2)|y = 1 \sim \mathcal{N}([2; 2], [5, 1; 1, 5])$. The z attribute has the Bernoulli distribution $p(z = 1) = p((x'_1, x'_2)|y = 1)/[p((x'_1, x'_2)|y = 0) + p((x'_1, x'_2)|y = 1)]$ where $(x'_1, x'_2) = (x_1 \cos(\pi/4) - x_2 \sin(\pi/4), x_1 \sin(\pi/4) + x_2 \cos(\pi/4))$.

We also use the real datasets ProPublica COMPAS (Angwin et al., 2016) and AdultCensus (Kohavi, 1996) datasets, which have 5,278 and 43,131 examples, respectively. We use the same pre-processing as in IBM's AI Fairness 360 (Bellamy et al., 2018) and use GENDER as the sensitive attribute. We also use the UTKFace dataset (Zhang et al., 2017) of 23,708 images, but only in Section 5.4.

Methods Compared For pre-processing unfairness mitigation methods, we propose a simple baseline called Cutting, which truncates the data to improve fairness by equalizing the amount of data within (y, z) classes or z classes, respectively. We can also oversample data instead of truncating, but the performance is not significantly different in our experiments. In addition, we compare with the state-of-the-art methods reweighing (Kamiran & Calders, 2011) and Label Bias Correction (Jiang & Nachum, 2020) (LBC). The reweighing algorithm applies weights on the examples to balance the importance between sensitive attributes. LBC finds example weights by repeatedly training the entire model to obtain an unbiased data distribution.

For in-processing methods, we compare with Fairness Constraints (Zafar et al., 2017a;b) (FC), Adversarial Debiasing (Zhang et al., 2018) (AD), and AdaFair (Iosifidis & Ntoutsi, 2019). FC applies a penalty term to the loss objective to reduce the difference in predictions between sensitive attributes. AD is an adversarial learning-based approach that maximizes the independence between predicted labels and the sensitive attributes. We improve AD's stability by not using the projection term. AdaFair is an AdaBoost-based fair algorithm that only applies to EO and requires multiple models to achieve fair results through a model ensemble. AdaFair gives more weights on the data points considered to be discriminated against, and it is the most similar approach in spirit to FairBatch. However, it is a boosting technique, which requires multiple model trainings.

Other Settings For the model architectures, we use logistic regression for all algorithms in Sections 5.2, 5.3, and 5.5 and utilize ResNet18 (He et al., 2016) and GoogLeNet (Szegedy et al., 2015) in Section 5.4. All the training is conducted through using the Adam optimizer. We perform the cross-validation on the train data to find the best hyper-parameters for each algorithm. The default α value is 0.1 throughout the experiments, and the batch sizes used for the synthetic, COMPAS, AdultCensus, and UTKFace datasets are 100, 200, 1,000, and 256, respectively. We use separate test data, and the ratios of the train versus test data for the synthetic and real data are 2:1 and 4:1, respectively. We repeat all experiments with ten different random seeds. We use PyTorch, and our experiments are performed on a server with Intel i7-6850 CPUs and NVIDIA TITAN Xp GPUs.

5.1 Results for Equal Opportunity

5.2 Results for Equalized Odds

Table 2 shows the accuracy and fairness of the algorithms on the synthetic, COMPAS, and AdultCensus test datasets w.r.t. equalized odds. The cutting and reweighing algorithms make fairer results than LR, but it does not achieve the fairness level of LBC, in-processing techniques, and FairBatch. In the AdultCensus dataset, both accuracy and fairness values of the cutting and the reweighing algorithms are worse than those of LR. It shows that some pre-processing algorithms that assumes a fully black-box model may result in unpredicted consequences as discussed in (Zafar et al., 2017a). Also, because the cutting algorithm discards information in training data, the accuracy even decreases than other fair algorithms. On the other hand, LBC and the in-processing techniques, FC, AD, and AdaFair, obtain fairer results than LR and simple pre-processing techniques. In comparison, FairBatch achieves better fairness than the best baselines on the COMPAS dataset, and it also has a similar fair level on the synthetic and AdultCensus datasets. In the supplementary, we also compare the change in AdaFair's data weights with the change in the FairBatch's class ratio. The result shows that AdaFair and FairBatch adjust the weight and ratio in a similar direction.

The number of epochs in Table 2 and wall clock times in Table 4 show the training speed of each algorithm. We observe that FairBatch has comparable speeds with simple baselines such as LR, Cutting, Reweighting, and FC. We now take a look at LBC and AdaFair, which are more relevant to FairBatch. LBC is as general as FairBatch because it requires minimal modification in the ML pipeline, and AdaFair is the most similar approach to FairBatch which gives more weight to discriminated examples. However, FairBatch is much faster than LBC (7-90 times in epochs, xx times in seconds) and AdaFair (17-96 times in epochs, xx times in seconds) because these require repetitive model training.

5.3 Results for Demographic Parity

We observe the performances of the algorithms w.r.t. demographic parity on the synthetic, COMPAS, and AdultCensus datasets in Table 3. The reweighing algorithm makes a fairer result than LR, but

Table 1: Accuracy and fairness performances on the synthetic, COMPAS, and AdultCensus test datasets w.r.t. equal opportunity (EP). We compare FairBatch with three types of baselines: (1) non-fair method: LR; (2) fair training via pre-processing: Cutting, Reweighing (Kamiran & Calders, 2011), and LBC (Jiang & Nachum, 2020); (3) fair training via in-processing: FC (Zafar et al., 2017b), AD (Zhang et al., 2018), and AdaFair (Iosifidis & Ntoutsi, 2019). The batch sizes are 100, 200, and 1000 for the synthetic, COMPAS, and AdultCensus datasets, respectively. We repeat all experiments with ten different random seeds.

	Synthetic			COMPAS			AdultCensus		
Method	Acc.	EP	Epochs	Acc.	EP	Epochs	Acc.	EP	Epochs
LR	.885±.000	.115±.000	400	.681±.002	.239±.006	300	.845±.001	.054±.005	300
Cutting	.879±.001	.085±.001	500	.533±.001	.188±.022	250	.809±.002	.266±.004	500
Reweighing	.858±.000	.020±.000	800	.685±.000	.137±.000	300	.835±.001	.134±.006	100
LBC	.858±.001	.022±.000	11200	.673±.002	.031±.006	3900	.841±.003	.011±.003	6300
FC	.833±.001	.007±.000	700	.656±.006	.059±.028	100	.844±.001	.021±.004	300
AD	.837±.010	.026±.007	200	.683±.001	.067±.029	300	.841±.003	.016±.005	400
AdaFair	.868±.000	.043±.001	16000	.664±.004	.018±.004	9600	.844±.001	.038±.004	9000
FairBatch	.844±.000	.007±.001	850	.682±.001	.028±.005	100	.845±.001	.030±.004	250

Table 2: Accuracy and fairness performances on the synthetic, COMPAS, and AdultCensus test datasets w.r.t. equalized odds (EO). Other settings are identical to Table 1.

	Synthetic			COMPAS			AdultCensus		
Method	Acc.	EO	Epochs	Acc.	EO	Epochs	Acc.	EO	Epochs
LR	.885±.000	.115±.000	400	.681±.002	.239±.006	300	.845±.001	.056±.003	300
Cutting	.859±.001	.036±.004	650	.665±.005	.066±.018	400	.802±.001	.062±.005	600
Reweighing	.856±.000	.038±.002	350	.685±.000	.137±.000	300	.835±.001	.134±.006	100
LBC	.858±.001	.026±.000	8800	.673±.002	.063±.005	9000	.840±.002	.027±.004	3300
FC	.865±.000	.030±.001	800	.677±.004	.101±.024	50	.841±.001	.038±.003	300
AD	.857±.000	.030±.001	1200	.683±.000	.082±.027	450	.843±.002	.033±.002	500
AdaFair	.868±.001	.029±.002	22400	.675±.000	.066±.002	9600	.843±.001	.038±.002	7800
FairBatch	.857±.001	.030±.001	850	.685±.001	.041±.004	100	.843±.000	.035±.002	450

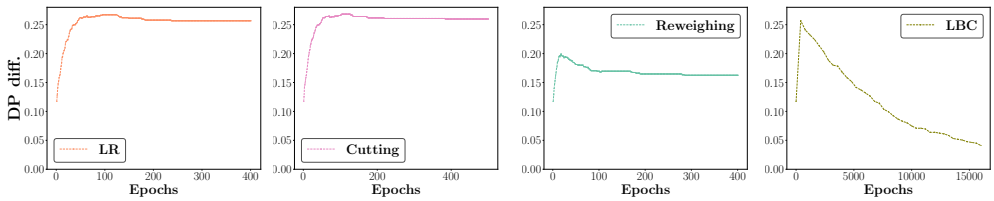
it does not achieve the fairness level of LBC, in-processing techniques, and FairBatch. Unlike the equalized odds results, the cutting algorithm has almost no improvement in demographic parity. This result shows that simply equalizing the data amount of each sensitive attribute is not useful to improve demographic parity. LBC and the in-processing techniques show similar results in Section 5.2. FairBatch again outperforms reweighing and cutting algorithms and makes comparable or better results to LBC and the in-processing techniques. Moreover, the accuracies of FairBatch are higher than LBC and in-processing techniques with similar fairness levels. Also, FairBatch is even faster than LBC (31-39 times in epochs, xx times in seconds) or AdaFair (31-104 times in epochs, xx times in seconds).

5.4 FairBatch for Fine-tuning the Models

While the previous experiments used a logistic regression model, in this section we show how FairBatch can improve the fairness of any unfair model. Table 5 shows how FairBatch performs for fine-tuning the pre-trained models ResNet18 (He et al., 2016) and GoogLeNet (Szegedy et al., 2015) on the UTKFace dataset. We observe the differences when using the original batch selection, the batch selection with cutting algorithm, and FairBatch for each pre-defined model. It shows that FairBatch is very effective for training the pre-trained models. [[More details after the experiment]]

Table 3: Accuracy and fairness performances on the synthetic, COMPAS, and AdultCensus test datasets w.r.t. demographic parity (DP). For DP, the baseline FC follows the algorithm Zafar et al. (2017a). Other settings are identical to Table 1.

	Synthetic			COMPAS			AdultCensus		
Method	Acc.	DP	Epochs	Acc.	DP	Epochs	Acc.	DP	Epochs
LR	.885±.000	.257±.000	400	.681±.002	.192±.006	300	.845±.001	.125±.001	300
Cutting	.885±.001	.258±.001	500	.677±.004	.205±.025	400	.846±.001	.123±.002	300
Reweighing	.857±.000	.164±.001	400	.685±.000	.103±.000	300	.835±.001	.052±.003	300
LBC	.768±.000	.042±.001	16000	.671±.002	.032±.009	7800	.815±.003	.011±.002	12600
FC	.785±.013	.058±.010	600	.684±.001	.083±.015	70	.812±.009	.025±.006	100
AD	.812±.008	.063±.014	700	.683±.002	.054±.019	550	.815±.008	.018±.004	400
AdaFair	.784±.001	.089±.001	52000	.642±.004	.033±.011	6300	.825±.002	.040±.001	27000
FairBatch	.800±.002	.042±.002	500	.683±.000	.043±.016	200	.825±.001	.010±.002	400



(a) LR (b) Cutting (c) Reweighing (d) LBC

(e) FC (f) AD (g) AdaFair (h) FairBatch

Figure 3: Epochs-DP violation curves of algorithms on the synthetic data.

5.5 FairBatch with Batch Selection Speed-up Strategies

FairBatch can be used with batch selection speed-up techniques proposed in general machine learning. These speed-up techniques generally assign different weights to each data example so that more important examples have higher sampling probabilities. We can gracefully merge FairBatch and speed-up techniques by applying sampling weights for each class within the amount of data specified by FairBatch. For example, combining FairBatch with loss-based weighting (Loshchilov & Hutter, 2016), which is a well-known batch selection speed-up technique, reduces the number of required epochs in FairBatch by about 50 epoch w.r.t. demographic parity on the synthetic data. We leave performance curves of FairBatch with/without loss-based weighting in the supplementary.

6 Related Work

Model Fairness Legal and social discussions translate into mathematical metrics to quantify the concepts of fairness (Narayanan, 2018). Among others, we focus on the main group fairness measures: equal opportunity (Hardt et al., 2016), equalized odds (Hardt et al., 2016), and demographic parity (Feldman et al., 2015). Unfairness mitigation techniques for group fairness have been proposed and can be categorized as follows: (1) pre-processing techniques (Kamiran & Calders, 2011; Zemel et al., 2013; Feldman et al., 2015; du Pin Calmon et al., 2017; Choi et al., 2020; Jiang & Nachum, 2020) that fix or generate the data, (2) in-processing techniques (Kamishima et al., 2012; Zafar et al., 2017a;b; Agarwal et al., 2018; Zhang et al., 2018; Cotter et al., 2019; Roh et al., 2020) that control the model training, and (3) post-processing techniques (Kamiran et al., 2012; Hardt et al., 2016; Pleiss

Table 4: Wall clock times (in seconds) for the training on the synthetic, COMPAS, and AdultCensus train datasets. The wall clock times are measured with the same settings in Tables 2 and 3.

Dataset	Target	LR	Cutting	Reweighing	LBC	FC	AD	AdaFair	FairBatch
Synthetic	EO	5.71	4.77	7.60	149.43	16.81	31.32	427.10	12.91
	DP	5.71	6.01	8.63	276.22	11.75	12.73	997.00	7.22
COMPAS	EO	—	—	—	—	—	—	—	—
	DP	—	—	—	—	—	—	—	—
AdultCensus	EO	—	—	—	—	—	—	—	—
	DP	—	—	—	—	—	—	—	—

Table 5: Accuracy and fairness performances of the pre-trained models with fine-tuning on the UTKFace test dataset w.r.t. equalized odds.

		(RACE, GENDER)			(RACE, AGE)		
Pre-trained model	Method	Acc.	EO	Epochs	Acc.	EO	Epochs
ResNet18	Original	.893±.002	.086±.012	19	.722±.011	.311±.053	10
	Cutting	.592±.020	.099±.014	18	.466±.018	.139±.021	20
	FairBatch	.891±.003	.065±.008	19	.716±.027	.186±.017	6
GoogLeNet	Original	.888±.003	.105±.016	20	.746±.006	.294±.034	14
	Cutting	.606±.010	.076±.017	20	.495±.017	.168±.033	9
	FairBatch	.000±.000	.000±.000	—	.000±.000	.000±.000	—

et al., 2017; Chzhen et al., 2019) that modify the outputs of the trained model. Most of these methods require significant engineering on major components of the deep learning pipeline and are difficult to perform without fairness expertise. In comparison, FairBatch is a batch selection technique that requires minimal change in code, runs in a single model training, and has competitive performance.

Although not our immediate focus, there are other noteworthy fairness measures: (1) individual fairness (Dwork et al., 2012) where close examples should be treated similarly, (2) causality-based fairness (Kilbertus et al., 2017; Kusner et al., 2017; Zhang & Bareinboim, 2018; Nabi & Shpitser, 2018; Khademi et al., 2019), which aims to overcome the limitations of non-causal approaches by analyzing the causal relationship surrounding the unfairness, and (3) distributionally robust optimization (DRO) (Sinha et al., 2017)-based fairness (Hashimoto et al., 2018), which achieves accuracy parity without the knowledge of sensitive attribute by balancing the risks across all distributions. Extending FairBatch to support these measures is an interesting future work.

Batch Selection Batch selection is an essential component in deep learning pipelines that use SGD, and there are studies that analyze the effect of batch sizes (Keskar et al., 2017; Masters & Luschi, 2018) as well as how to perform batch sampling (Shamir, 2016; Gürbüzbalaban et al., 2019). More recently, various importance sampling techniques have been proposed to select more useful batches and thus accelerate model training (Loshchilov & Hutter, 2016; Alain et al., 2016; Stich et al., 2017; Csiba & Richtárik, 2018; Katharopoulos & Fleuret, 2018; Johnson & Guestrin, 2018). In comparison, FairBatch makes the novel contribution of improving model fairness through batch selection and can be integrated with existing importance sampling techniques.

Hyperparameter Optimization

7 Conclusion

We proposed FairBatch, which is a batch selection technique for mini-batch SGD that can be used to improve model fairness efficiently and easily. Our key contribution is formalizing a bilevel optimization problem and proposing a normalized gradient descent algorithm that is proven to converge. We then propose an efficient approximation algorithm that runs within a single model training by adjusting sensitive group ratios within each batch. We showed how FairBatch supports

equal opportunity and can be extended to support equalized odds and demographic parity. In our experiments, FairBatch results in fairness and accuracy results that are on par with state-of-the-art pre-processing and in-processing unfairness mitigation techniques. In addition, FairBatch is not only efficient, but easily applicable to non-fair models and compatible with existing importance sampling techniques for faster convergence.

REFERENCES

Alekh Agarwal, Alina Beygelzimer, Miroslav Dudík, John Langford, and Hanna M. Wallach. A reductions approach to fair classification. In *ICML*, pp. 60–69, 2018.

Guillaume Alain, Alex Lamb, Chinnadhurai Sankar, Aaron Courville, and Yoshua Bengio. Variance reduction in sgd by distributed importance sampling. *ICLR Workshop*, 2016.

J. Angwin, J. Larson, S. Mattu, and L. Kirchner. Machine bias: There's software used across the country to predict future criminals. And its biased against blacks., 2016.

Rachel K. E. Bellamy, Kuntal Dey, Michael Hind, Samuel C. Hoffman, Stephanie Houde, Kalapriya Kannan, Pranay Lohia, Jacquelyn Martino, Sameep Mehta, Aleksandra Mojsilovic, Seema Nagar, Karthikeyan Natesan Ramamurthy, John Richards, Diptikalyan Saha, Prasanna Sattigeri, Moninder Singh, Kush R. Varshney, and Yunfeng Zhang. AI Fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias, October 2018. URL `https://arxiv.org/abs/1810.01943`.

Kristy Choi, Aditya Grover, Trisha Singh, Rui Shu, and Stefano Ermon. Fair generative modeling via weak supervision. In *ICML*, 2020.

Evgenii Chzhen, Christophe Denis, Mohamed Hebiri, Luca Oneto, and Massimiliano Pontil. Leveraging labeled and unlabeled data for consistent fair binary classification. In *NeurIPS*, pp. 12760–12770. 2019.

Andrew Cotter, Heinrich Jiang, and Karthik Sridharan. Two-player games for efficient non-convex constrained optimization. In *ALT*, pp. 300–332, 2019.

Dominik Csiba and Peter Richtárik. Importance sampling for minibatches. *The Journal of Machine Learning Research*, 19(1):962–982, 2018.

Flávio du Pin Calmon, Dennis Wei, Bhanukiran Vinzamuri, Karthikeyan Natesan Ramamurthy, and Kush R. Varshney. Optimized pre-processing for discrimination prevention. In *NeurIPS*, pp. 3995–4004, 2017.

Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard S. Zemel. Fairness through awareness. In *ITCS*, pp. 214–226, 2012. doi: 10.1145/2090236.2090255.

Michael Feldman, Sorelle A. Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. Certifying and removing disparate impact. In *KDD*, pp. 259–268, 2015. doi: 10.1145/2783258.2783311.

Mert Gürbüzbalaban, Asu Ozdaglar, and PA Parrilo. Why random reshuffling beats stochastic gradient descent. *Mathematical Programming*, pp. 1–36, 2019.

Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. In *NeurIPS*, pp. 3315–3323, 2016.

Tatsunori Hashimoto, Megha Srivastava, Hongseok Namkoong, and Percy Liang. Fairness without demographics in repeated loss minimization. In *ICML*, pp. 1929–1938, 2018.

Elad Hazan, Kfir Y. Levy, and Shai Shalev-Shwartz. Beyond convexity: Stochastic quasi-convex optimization. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 1594–1602, 2015. URL `http://papers.nips.cc/paper/5718-beyond-convexity-stochastic-quasi-convex-optimization`.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CVPR)*, Jun 2016.

Vasileios Iosifidis and Eirini Ntoutsi. Adafair: Cumulative fairness adaptive boosting. In *CIKM*, pp. 781–790, 2019.

Heinrich Jiang and Ofir Nachum. Identifying and correcting label bias in machine learning. In *AISTATS*, pp. 702–712, 2020.

Tyler B Johnson and Carlos Guestrin. Training deep models faster with robust, approximate importance sampling. In *NIPS*, pp. 7265–7275, 2018.

Faisal Kamiran and Toon Calders. Data preprocessing techniques for classification without discrimination. *Knowl. Inf. Syst.*, 33(1):1–33, 2011. doi: 10.1007/s10115-011-0463-8.

Faisal Kamiran, Asim Karim, and Xiangliang Zhang. Decision theory for discrimination-aware classification. In *ICDM*, pp. 924–929, 2012. doi: 10.1109/ICDM.2012.45.

Toshihiro Kamishima, Shotaro Akaho, Hideki Asoh, and Jun Sakuma. Fairness-aware classifier with prejudice remover regularizer. In *ECML PKDD*, pp. 35–50, 2012. doi: 10.1007/978-3-642-33486-3_3.

Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *ICML*, pp. 2530–2539, 2018.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *ICLR*, 2017.

Aria Khademi, Sanghack Lee, David Foley, and Vasant Honavar. Fairness in algorithmic decision making: An excursion through the lens of causality. In *WWW*, pp. 2907—-2914, 2019.

Niki Kilbertus, Mateo Rojas-Carulla, Giambattista Parascandolo, Moritz Hardt, Dominik Janzing, and Bernhard Schölkopf. Avoiding discrimination through causal reasoning. In *NeurIPS*, pp. 656—-666, 2017.

Ron Kohavi. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *KDD*, pp. 202–207, 1996.

Matt J Kusner, Joshua Loftus, Chris Russell, and Ricardo Silva. Counterfactual fairness. In *NeurIPS*, pp. 4066–4076. 2017.

I. Loshchilov and F. Hutter. Online batch selection for faster training of neural networks. In *ICLR 2016 Workshop Track*, 2016.

Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks. *arXiv preprint arXiv:1804.07612*, 2018.

Razieh Nabi and Ilya Shpitser. Fair inference on outcomes. *AAAI*, pp. 1931–1940, 2018.

Arvind Narayanan. Translation tutorial: 21 fairness definitions and their politics. In *Proc. Conf. Fairness Accountability Transp., New York, USA*, volume 1170, 2018.

Y. E. Nesterov. Minimization methods for nonsmooth convex and quasiconvex functions. *Matekon)*, 1984.

Geoff Pleiss, Manish Raghavan, Felix Wu, Jon M. Kleinberg, and Kilian Q. Weinberger. On fairness and calibration. In *NeurIPS*, pp. 5684–5693, 2017.

Yuji Roh, Kangwook Lee, Steven Euijong Whang, and Changho Suh. FR-Train: A mutual information-based approach to fair and robust training. *ICML*, 2020.

Ohad Shamir. Without-replacement sampling for stochastic gradient methods. In *NIPS*, pp. 46–54. 2016.

Aman Sinha, Hongseok Namkoong, and John C. Duchi. Certifying some distributional robustness with principled adversarial training. In *ICLR*, 2017.

Sebastian U Stich, Anant Raj, and Martin Jaggi. Safe adaptive importance sampling. In *NIPS*, pp. 4381–4391, 2017.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CVPR*, Jun 2015.

Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez-Rodriguez, and Krishna P. Gummadi. Fairness constraints: Mechanisms for fair classification. In *AISTATS*, pp. 962–970, 2017a.

Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez-Rodriguez, and Krishna P. Gummadi. Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment. In *WWW*, pp. 1171–1180, 2017b.

Richard S. Zemel, Yu Wu, Kevin Swersky, Toniann Pitassi, and Cynthia Dwork. Learning fair representations. In *ICML*, pp. 325–333, 2013.

Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. Mitigating unwanted biases with adversarial learning. In *AIES*, pp. 335–340, 2018. doi: 10.1145/3278721.3278779.

Junzhe Zhang and Elias Bareinboim. Fairness in decision-making - the causal explanation formula. In *AAAI*, 2018.

Zhifei Zhang, Yang Song, , and Hairong Qi. Age progression/regression by conditional adversarial autoencoder. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017.

A APPENDIX

Appendix A.2 proves Lemma 1. Appendix A.4 shows the extended algorithms for equalized odds and demographic parity. Appendix A.5 shows the convergence of FairBatch when also using importance sampling.

A.1 NUMERICAL EXAMPLES

$f(\cdot)$ and $g(\cdot)$ are convex, but $F(\lambda)$ is not convex.

$$f(x) = \frac{e^x + e^{-x}}{5}, \;\; g(x) = (x-1)^2$$

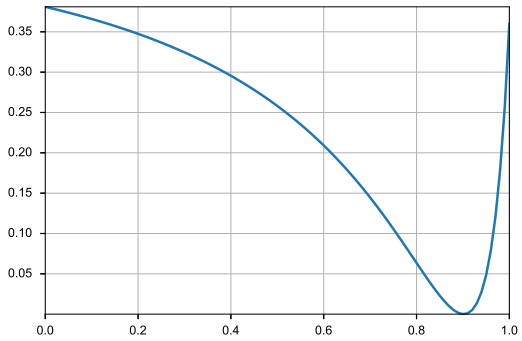


Figure 4: $F(\lambda)$ is not convex but quasi-convex.

A.2 PROOF FOR LEMMA 1

In this section, we provide a formal statement of Lemma 1 and prove it.

Lemma 3 (Quasi-convexity of $F(\lambda)$)**.** *If $f(\cdot)$, $g(\cdot)$ and $h(\cdot)$ satisfy*

1. $h(x) = 0$ *or*
2. $\lambda \nabla^2 f(x^*_\lambda) + (1-\lambda)\nabla^2 g(x^*_\lambda) + \nabla^2 h(x^*_\lambda) \succ 0$ *for every* $\lambda \in [0,1]$,

then the outer objective function $F(\lambda)$ is quasi-convex. Also, sign $\left(\frac{\mathrm{d}F(\lambda)}{\mathrm{d}\lambda}\right) = \mathbb{1}(f(x^*_\lambda) \leq g(x^*_\lambda))$.

Proof. It it known that a continuous function $f : \mathbb{R} \to \mathbb{R}$ is quasiconvex if and only if at least one of the following conditions holds: 1) nondecreasing, 2) nonincreasing, and 3) nonincreasing and then nondecreasing [Boyd04]. We will prove the lemma by showing that the function $F(\lambda)$ is quasiconvex by showing that it is nonincreasing and then nondecreasing. More precisely, we will show that $f(x^\star_\lambda) - g(x^\star_\lambda)$ is a nondecreasing function. It is easy to see that this directly implies that $(f(x^\star_\lambda) - g(x^\star_\lambda))^2$ is nonincreasing and then nondecreasing.

Case 1 ($h(x) = 0$) Consider λ_1 and λ_2 such that $\lambda_1 > \lambda_2$. If we can show $f(x^\star_{\lambda_1}) \leq f(x^\star_{\lambda_2})$ and $g(x^\star_{\lambda_1}) \geq g(x^\star_{\lambda_2})$, then this implies that $f(x^\star_\lambda) - g(x^\star_\lambda)$ is a nondecreasing function. Indeed, this is very intuitive: If we increase λ, the inner optimization problems puts a higher weight on $f(\cdot)$, resulting in a lower value of $f(x^\star)$ and a higher value of $g(x^\star)$. We formally show this by contradiction. By the definition of $x^\star_\lambda$, we have the following two conditions:

$$\lambda_1 f(x^\star_{\lambda_1}) + (1-\lambda_1) g(x^\star_{\lambda_1}) \leq \lambda_1 f(x) + (1-\lambda_1) g(x), \; \forall x, \tag{5}$$

$$\lambda_2 f(x^\star_{\lambda_2}) + (1-\lambda_2) g(x^\star_{\lambda_2}) \leq \lambda_2 f(x) + (1-\lambda_2) g(x), \; \forall x. \tag{6}$$

If the lemma's statement is false, one of the following three events should occur:

1. $f(x^\star_{\lambda_1}) > f(x^\star_{\lambda_2})$ and $g(x^\star_{\lambda_1}) \geq g(x^\star_{\lambda_2})$: By adding these two inequalities with respective weights λ_1 and $1-\lambda_1$, we have $\lambda_1 f(x^\star_{\lambda_1}) + (1-\lambda_1) g(x^\star_{\lambda_1}) > \lambda_1 f(x^\star_{\lambda_2}) + (1-\lambda_1) g(x^\star_{\lambda_2})$. This contradicts equation 5.

2. $f(x^\star_{\lambda_1}) \leq f(x^\star_{\lambda_2})$ and $g(x^\star_{\lambda_1}) < g(x^\star_{\lambda_2})$: Similarly, by adding these two inequalities with respective weights λ_2 and $1-\lambda_2$, we have $\lambda_2 f(x^\star_{\lambda_2}) + (1-\lambda_2) g(x^\star_{\lambda_2}) > \lambda_2 f(x^\star_{\lambda_1}) + (1-\lambda_2) g(x^\star_{\lambda_1})$. This contradicts equation 6.

3. $f(x^\star_{\lambda_1}) > f(x^\star_{\lambda_2})$ and $g(x^\star_{\lambda_1}) < g(x^\star_{\lambda_2})$: By adding equation 5 with $x = x^\star_{\lambda_2}$ and equation 6 with $x = x^\star_{\lambda_1}$, we have

$$\begin{aligned}&\lambda_1 f(x^\star_{\lambda_1}) + (1-\lambda_1) g(x^\star_{\lambda_1}) + \lambda_2 f(x^\star_{\lambda_2}) + (1-\lambda_2) g(x^\star_{\lambda_2}) \\ &\leq \lambda_1 f(x^\star_{\lambda_2}) + (1-\lambda_1) g(x^\star_{\lambda_2}) + \lambda_2 f(x^\star_{\lambda_1}) + (1-\lambda_2) g(x^\star_{\lambda_1}).\end{aligned}$$

   By rearranging terms, we have

$$(\lambda_1 - \lambda_2)(f(x^\star_{\lambda_1}) - f(x^\star_{\lambda_2})) \leq (\lambda_1 - \lambda_2)(g(x^\star_{\lambda_1}) - g(x^\star_{\lambda_2})).$$

   By dividing both sides by $\lambda_1 - \lambda_2 > 0$, we have $f(x^\star_{\lambda_1}) - f(x^\star_{\lambda_2}) \leq g(x^\star_{\lambda_1}) - g(x^\star_{\lambda_2})$. This contradicts the condition as the left-hand side is positive while the right-hand side is negative.

This completes the proof of the first claim by contradiction.

The second claim immediately follows the first claim. Since $F(\lambda) = \frac{1}{2}(f(x^\star_\lambda) - g(x^\star_\lambda))^2$, we have $\frac{\mathrm{d}F(\lambda)}{\mathrm{d}\lambda} = (f(x^\star_\lambda) - g(x^\star_\lambda)) \frac{\mathrm{d}}{\mathrm{d}\lambda}(f(x^\star_\lambda) - g(x^\star_\lambda))$. As shown in the earlier part of this proof, we know $\frac{\mathrm{d}f(x^\star_\lambda)}{\mathrm{d}\lambda} \leq 0$ and $\frac{\mathrm{d}g(x^\star_\lambda)}{\mathrm{d}\lambda} \geq 0$. Thus, $\mathrm{sign}(\frac{\mathrm{d}F(\lambda)}{\mathrm{d}\lambda}) = \mathrm{sign}(g(x^\star_\lambda) - f(x^\star_\lambda)) = \mathbb{1}(f(x^\star_\lambda) \leq g(x^\star_\lambda))$.

Case 2 ($\lambda\nabla^2 f(x^*_\lambda) + (1-\lambda)\nabla^2 g(x^*_\lambda) + \nabla^2 h(x^*_\lambda) \succ 0$ for every $\lambda \in [0,1]$) In this part of the proof, we will denote $x^\star_\lambda$ by x for simplicity. To show that $f(x) - g(x)$ is a nondecreasing function (in λ), consider the derivative:

$$\frac{\mathrm{d}}{\mathrm{d}\lambda}(f(x) - g(x)) = (\nabla f(x) - \nabla g(x))^\top \frac{\mathrm{d}x}{\mathrm{d}\lambda} \tag{7}$$

To compute $\frac{\mathrm{d}x}{\mathrm{d}\lambda}$, we implicitly differentiate (with respect to λ) the following stationary equation.

$$\begin{aligned}&\lambda\nabla f(x) + (1-\lambda)\nabla g(x) + \nabla h(x) = 0 \\ \Rightarrow\quad &\nabla f(x) + \lambda\nabla^2 f(x) \cdot \frac{\mathrm{d}x}{\mathrm{d}\lambda} - \nabla g(x) + (1-\lambda)\nabla^2 g(x) \cdot \frac{\mathrm{d}x}{\mathrm{d}\lambda} + \nabla^2 h(x) \cdot \frac{\mathrm{d}x}{\mathrm{d}\lambda} = 0\end{aligned}$$

By rearranging terms, we have

$$\left(\lambda\nabla^2 f(x) + (1-\lambda)\nabla^2 g(x) + \nabla^2 h(x)\right) \frac{\mathrm{d}x}{\mathrm{d}\lambda} = -(\nabla f(x) - \nabla g(x)).$$

By the assumption, $\lambda\nabla^2 f(x) + (1-\lambda)\nabla^2 g(x) + \nabla^2 h(x)$ is positive definite and hence invertible. Thus,

$$\frac{\mathrm{d}x}{\mathrm{d}\lambda} = -\left(\lambda\nabla^2 f(x) + (1-\lambda)\nabla^2 g(x) + \nabla^2 h(x)\right)^{-1} (\nabla f(x) - \nabla g(x)).$$

Therefore,

$$\frac{\mathrm{d}}{\mathrm{d}\lambda}(f(x) - g(x)) = -(\nabla f(x) - \nabla g(x))^\top \left(\lambda\nabla^2 f(x) + (1-\lambda)\nabla^2 g(x) + \nabla^2 h(x)\right)^{-1} (\nabla f(x) - \nabla g(x)).$$

Note that $\left(\lambda\nabla^2 f(x) + (1-\lambda)\nabla^2 g(x) + \nabla^2 h(x)\right)^{-1}$ is also positive definite. Thus, $\frac{\mathrm{d}}{\mathrm{d}\lambda}(f(x) - g(x))$ is always negative, and hence $f(x) - g(x)$ is a decreasing function.

Now, observe that

$$\frac{\mathrm{d}F(\lambda)}{\mathrm{d}\lambda} = (f(x) - g(x)) \cdot \frac{\mathrm{d}}{\mathrm{d}\lambda}(f(x) - g(x)).$$

Therefore, $\mathrm{sign}\left(\frac{\mathrm{d}F(\lambda)}{\mathrm{d}\lambda}\right) = \mathbb{1}(f(x) \leq g(x))$. □

Algorithm 2: Adaptive adjustment of class ratio within batch w.r.t. equalized odds.

Input: Intermediate model, criterion, train data (x_{train}, z_{train}, y_{train}), previous lambda λ_{t-1}, and FairBatch's learning rate α

output = model (x_{train})

$loss$ = criterion (output, y_{train})

$$sign[(y,z)] = \begin{cases} +1, & \text{if mean}(loss[(y,z)]) > \text{mean}(loss[y]) \\ -1, & \text{if mean}(loss[(y,z)]) < \text{mean}(loss[y]) \\ 0, & \text{otherwise} \end{cases}$$

$$cr_t[(y,z)] = \begin{cases} cr_{t-1}[(y,z)] + \alpha, & \text{if sign}[(y,z)] > 0 \\ cr_{t-1}[(y,z)] - \alpha, & \text{if sign}[(y,z)] < 0 \\ cr_{t-1}[(y,z)], & \text{otherwise} \end{cases}$$

Output : Next lambda λ_t

Algorithm 3: Adaptive adjustment of class ratio within batch w.r.t. demographic parity.

Input: Intermediate model, criterion, train data (x_{train}, z_{train}, y_{train}), previous lambda λ_{t-1}, and FairBatch's learning rate α

output = model (x_{train})

$loss$ = criterion (output, $\mathbf{1}$)

$$(sign[(1,z)], sign[(0,z)]) = \begin{cases} (+1,-1), & \text{if mean}(loss[z]) > \text{mean}(loss) \\ (-1,+1), & \text{if mean}(loss[z]) < \text{mean}(loss) \\ (\ 0,\ \ 0), & \text{otherwise} \end{cases}$$

$$cr_t[(y,z)] = \begin{cases} cr_{t-1}[(y,z)] + \alpha, & \text{if sign}[(y,z)] > 0 \\ cr_{t-1}[(y,z)] - \alpha, & \text{if sign}[(y,z)] < 0 \\ cr_{t-1}[(y,z)], & \text{otherwise} \end{cases}$$

Output : Next lambda λ_t

A.3 MULTI-DIMENSIONAL λ

It is well known that the standard gradient descent method may not converge when applied to a quasi-convex function. To see this, consider an example where the gradients near the global optimum can be very large. To avoid this, one can use a *normalized* gradient instead, i.e., one should use $\frac{\nabla f}{\|\nabla f\|}$ instead of ∇f (Hazan et al., 2015). Therefore, if we can prove that $F(\lambda)$ is quasi-convex, we can optimize the bilevel optimization problem as follows:

$$\lambda^{(t+1)} = \lambda^{(t)} - \alpha_t \cdot \text{sign}\left(\left. \frac{dF(\lambda)}{d\lambda} \right|_{\lambda=\lambda^{(t)}} \right). \tag{8}$$

Note that λ is a scalar value here, so the normalization step is identical to applying the sign function to the derivative.

A.4 EQUALIZED ODDS AND DEMOGRAPHIC PARITY

We provide details of the extended FairBatch algorithms for equalized odds and demographic parity.

Equalized Odds Algorithm 2 shows how to adjust the class ratios per batch for equalized odds.

Demographic Parity Algorithm 3 shows how to adjust the class ratios per batch for demographic parity.

A.5 FAIRBATCH WITH IMPORTANCE SAMPLING

Figure 5 shows the convergence of FairBatch when combined with importance sampling.

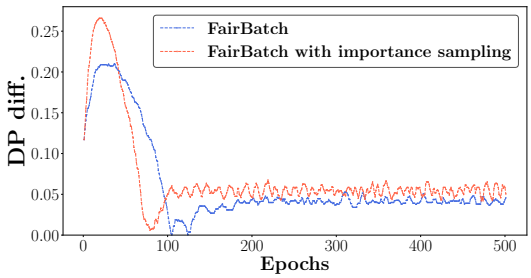


Figure 5: Epochs-DP violation curve of FairBatch on the synthetic dataset, with/without loss-based weighting (Loshchilov & Hutter, 2016).