Fair Infinitesimal Jackknife: Mitigating the Influence of Biased Training Data Points Without Refitting

Prasanna Sattigeri
IBM Research
Yorktown Heights, NY
psattig@us.ibm.com

Soumya Ghosh
MIT-IBM Watson AI Lab, IBM Research
Cambridge, MA
ghoshso@us.ibm.com

Inkit Padhi
IBM Research
Yorktown Heights, NY
inkit.padhi@ibm.com

Pierre Dognin
IBM Research
Yorktown Heights, NY
pdognin@us.ibm.com

Kush R. Varshney
IBM Research
Yorktown Heights, NY
krvarshn@us.ibm.com

Abstract

In consequential decision-making applications, mitigating unwanted biases in machine learning models that yield systematic disadvantage to members of groups delineated by sensitive attributes such as race and gender is one key intervention to strive for equity. Focusing on demographic parity and equality of opportunity, in this paper we propose an algorithm that improves the fairness of a pre-trained classifier by simply dropping carefully selected training data points. We select instances based on their influence on the fairness metric of interest, computed using an infinitesimal jackknife-based approach. The dropping of training points is done in principle, but in practice does not require the model to be refit. Crucially, we find that such an intervention does not substantially reduce the predictive performance of the model but drastically improves the fairness metric. Through careful experiments, we evaluate the effectiveness of the proposed approach on diverse tasks and find that it consistently improves upon existing alternatives.

1 Introduction

Among the many possible interventions to improve equity in society (most of them involve structural policy change), bias mitigation algorithms constitute one narrow sliver that has emerged in the machine learning literature to address distributive justice in high-stakes automated decision making. These algorithms may be categorized into pre-processing, in-processing, and post-processing approaches [37]. In the case of in-processing algorithms [20], the bias mitigation intervention occurs at the model training stage. This is usually achieved by minimizing the empirical risk regularized by a fairness metric surrogate that captures the dependence of the prediction and the sensitive attribute. Pre-processing methods typically learn transformations of the data distribution such that they do not contain information about the sensitive attributes [43, 26]. Task specific models are then learned from scratch on these debiased representations. Retraining a model from scratch is intractable in many real-world situations for a variety of reasons including policy, cost, and technical feasibility; post-processing approaches are the only viable option in such cases. For example, consider trying to refit large foundation models. Limiting ourselves to notions of *group* fairness such as demographic parity and equality of opportunity, existing post-processing bias mitigation algorithms tend to either randomly or deterministically alter the hard or soft predicted label of individual test data points that have been scored by a model [19, 17, 30, 8, 25, 39].

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

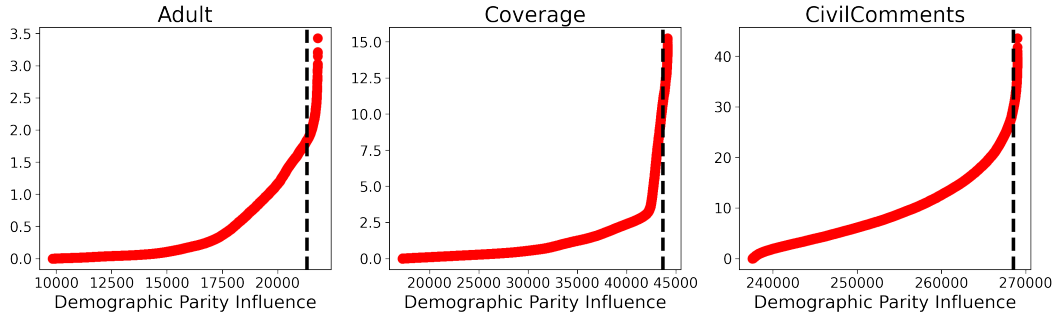

Figure 1: **Fairness influence**. We plot the sorted influence scores of training instances on the average demographic parity of a held-out validation set for different datasets considered in this paper. Only training instances with positive influence on demographic parity are plotted. The points to the right of the black line are the 500 most influential training instances. Models adjusted to mitigate the influence of these instances are substantially more fair. On held-out test sets of the *Adult*, *Coverage*, *CivilComments* datasets, demographic parity improves from 0.18 to 0.01, from 0.22 to 0.01, and 0.17 to 0.03.

In this paper, we propose a more "global" bias mitigation algorithm. Our procedure alters the entire model without a focus on individual *test* points. Similar to post-processing approaches, our method mitigates pre-trained models without requiring any additional refitting of the model. Unlike standard post-processing approaches, however, our method does require access to the training data. In exchange for this additional requirement, we find that our approach typically substantially outperforms other post-processing techniques and can even augment in-processing approaches for a better fairness/accuracy trade-off.

Our contributions. Our first contribution is methodological. We use the notion of influence functions to estimate the "influence" of training instances on various group fairness metrics of interest. We then perform post-hoc unfairness mitigation by approximately removing training instances that have a disproportional impact on group (un)fairness. We theoretically analyze the proposed approach and establish conditions under which it provably improves group fairness.

Next, we observe that influence calculations require the inversion of a Hessian matrix, a prohibitively expensive operation for models with a large number of parameters. Existing approximations [21, 1, 34] can either be expensive, inaccurate, or unstable [29, 4, 35]. We develop *IHVP-WoodFisher*, a WoodFisher [35] based Inverse-Hessian Vector Product (IHVP) scheme for computing the fairness influence score of the training instances that is stable, easy to compute, and does not require constraints, such as restricted Eigenspectrum of the loss curvature, that are hard to satisfy in practice.

Our final contribution is empirical. First, through careful experiments on tabular data, we show that our approach is effective at reducing group unfairness, is competitive with existing methods, and can even augment the latter to achieve a better fairness/accuracy trade-off. Then, we demonstrate how our approach can be easily adapted to more complex modalities such as natural language and be used for bias mitigation of large pre-trained language models through *prompt-tuning*, a use-case that is likely to become increasingly common with the proliferation of large language models.

2 Background and Related Work

2.1 Empirical and Weighted Risk Minimization

We begin by considering the standard supervised learning setup. Given a dataset $\mathcal{D} = \{\mathbf{z}_n = (\mathbf{x}_n, y_n)\}_{n=1}^N$ of N features ($\mathbf{x}_n \in \mathbb{R}^p$), response pairs ($y_n \in \mathcal{Y}$), a model $h_{\boldsymbol{\theta}}(\mathbf{x})$ parameterized by a set of parameters $\boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^D$, and a loss function $\ell : \Theta \times \mathcal{Y} \to \mathbb{R}$, we minimize the empirical risk,

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \Theta}{\operatorname{argmin}} \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, h_{\boldsymbol{\theta}}(\mathbf{x}_n)), \tag{1}$$

to arrive at a trained predictor $h_{\hat{\boldsymbol{\theta}}}(\mathbf{x})$. We will denote $L(\boldsymbol{\theta}) \stackrel{\text{def}}{=} \frac{1}{N}\sum_{n=1}^{N} \ell(y_n, h_{\boldsymbol{\theta}}(\mathbf{x}_n))$ for notational convenience. Next, consider a weighted risk minimization problem,

$$\hat{\boldsymbol{\theta}}(\mathbf{w}) = \underset{\boldsymbol{\theta}\in\Theta}{\operatorname{argmin}} \frac{1}{N}\sum_{n=1}^{N} w_n \ell(y_n, h_{\boldsymbol{\theta}}(\mathbf{x}_n)), \tag{2}$$

that weights the loss at each training instance by a scalar weight w_n and $\mathbf{w}$ denotes the column vector $[w_1, w_2, \ldots, w_N]^T \in \mathbb{R}^N$. Setting all the weights to one, $\mathbf{1} \stackrel{\text{def}}{=} [w_1 = 1, w_2 = 1, \ldots, w_N = 1]^T$, and minimizing the right hand side of Equation 2 recovers the standard empirical risk minimization problem. On the other hand, setting the nth coordinate to zero recovers the solution to an empirical risk minimization problem after dropping the nth training instance. As is clear from Equation 2 and emphasized by our notation, $\boldsymbol{\theta}$ is a function of the weights $\mathbf{w}$. Although we typically do not have a closed form expression for this function, we can form a Taylor approximation to it:

$$\boldsymbol{\theta}(\mathbf{w}) = \hat{\boldsymbol{\theta}} + \nabla_{\mathbf{w}}\boldsymbol{\theta}(\mathbf{w})\Big|_{\mathbf{w}=\mathbf{1}} (\mathbf{w} - \mathbf{1}) + \mathcal{O}((\mathbf{w} - \mathbf{1})^2), \tag{3}$$

where $\nabla_{\mathbf{w}}\boldsymbol{\theta}(\mathbf{w}) \in \mathbb{R}^{D\times N}$ is the Jacobian matrix. This first order Taylor approximation is often referred to as the infinitesimal jackknife approximation [18, 14]. The coordinate-wise gradient $\frac{d\theta(\mathbf{w})}{dw_n}|_{w_n=1}$ measures the effect of perturbing the weight of the nth data point on $\hat{\boldsymbol{\theta}}$ and is popularly referred to as the influence function [21], since it measures the "influence" of the nth training instance on the model's parameters. When we are at a stationary point of $L(\boldsymbol{\theta})$, i.e., when $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = 0$, $L(\boldsymbol{\theta})$ is twice differentiable in $\boldsymbol{\theta}$, then,

$$\frac{d\theta(\mathbf{w})}{dw_n}\Big|_{\mathbf{w}=\mathbf{1}} = -H^{-1} g_n, \tag{4}$$

where $H \stackrel{\text{def}}{=} \nabla^2_{\boldsymbol{\theta}} L(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}}$, and $g_n \stackrel{\text{def}}{=} \nabla_{\boldsymbol{\theta}} \ell(y_n, h_{\boldsymbol{\theta}}(\mathbf{x}_n))|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}}$. Recent work [15] has shown that the above expression approximates the gradient well in the vicinity of a stationary point with the accuracy of the approximation deteriorating smoothly with increasing distance from the stationary point. This result justifies the use of influence functions even when stochastic optimization is used for minimizing Equation 1. Finally, to measure the influence of a training instance on a differentiable functional, M, of $\boldsymbol{\theta}(\mathbf{w})$, we apply chain rule to arrive at,

$$\mathcal{I}_{M,n} \stackrel{\text{def}}{=} \frac{dM(\boldsymbol{\theta}(\mathbf{w}), \mathbf{w})}{dw_n}\Big|_{\mathbf{w}=\mathbf{1}, \boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} = -\nabla_{\boldsymbol{\theta}} M(\boldsymbol{\theta}(\mathbf{w}), \mathbf{w})\Big|^{T}_{\mathbf{w}=\mathbf{1}, \boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} H^{-1} g_n, \tag{5}$$

where our notation makes explicit the dependence of M on $\mathbf{w}$. Recent work has leveraged this machinery to approximate cross-validation [16, 36, 15], to interpret black-box machine learning models [21], and to assess the sensitivity of statistical analyses to training data perturbations [6], among others. Differently from these, we show how this machinery can be leveraged for reducing disparities of pre-trained models across groups.

2.2 Fair Classification

We further assume that for each data instance we have access to a sensitive attribute $\mathbf{s}_n \in [k]$, i.e., $\mathcal{D} = \{\mathbf{z}_n = (\mathbf{x}_n, s_n, y_n)\}_{n=1}^N$, that encodes the protected group membership of the nth data instance and that we are interested in binary classification, $\mathcal{Y} = \{1, 0\}$. In fair classification, we want to learn accurate classifiers that minimize disparities in predictions across groups.

To quantify disparities across groups, we primarily focus on two common fairness metrics — demographic (or statistical) parity (DP) [3] and equality of odds (EO) [17]. DP requires the classifier's predictions to be statistically independent of the sensitive attribute, $h_{\boldsymbol{\theta}}(X) \perp\!\!\!\perp S$, where X and S are random variables representing the features and the sensitive attribute. For a binary sensitive attribute, DP implies, $P(h_{\boldsymbol{\theta}}(X) = 1 \mid S = 1) = P(h_{\boldsymbol{\theta}}(X) = 1 \mid S = 0)$. EO, on the other hand, requires the classifier's predictions to be statistically independent of the sensitive attribute *conditioned* on the true outcome, $h_{\boldsymbol{\theta}}(X) \perp\!\!\!\perp S \mid Y$. For a binary sensitive attribute, EO implies $P(h_{\boldsymbol{\theta}}(X) = 1 \mid S = 1, Y = y) = P(h_{\boldsymbol{\theta}}(X) = 1 \mid S = 0, Y = y)$ for both $y = 0$ and $y = 1$. Equality of opportunity (EQOPP) [17] is a special case of equality of odds where the predictions are conditionally independent of the sensitive attribute given the true outcome is positive. A common

strategy for learning fair classifiers is to then require the absolute difference in demographic parity (DP),

$$\Delta\text{DP}(\boldsymbol{\theta}) = |P(h_{\boldsymbol{\theta}}(X) = 1 \mid S = 1) - P(h_{\boldsymbol{\theta}}(X) = 1 \mid S = 0)|,$$

or the absolute difference in equality of odds,

$$\Delta\text{EO}(\boldsymbol{\theta}) = \sum_{y=0}^{1} |P(h_{\boldsymbol{\theta}}(X) = 1 \mid S = 1, Y = y) - P(h_{\boldsymbol{\theta}}(X) = 1 \mid S = 0, Y = y)|,$$

to be close to zero while minimizing the empirical risk (Equation 1). Smooth[1] surrogates to ΔDP and ΔEO that are estimated from an empirical distribution are commonly used in practice [42],

$$\begin{aligned} M_{\mathcal{D}}^{\Delta\text{DP}}(\boldsymbol{\theta}) &= |\mathbb{E}_{p_{\mathcal{D}}(X=\mathbf{x}|S=1)}[h_{\boldsymbol{\theta}}(\mathbf{x})] - \mathbb{E}_{p_{\mathcal{D}}(X=\mathbf{x}|S=0)}[h_{\boldsymbol{\theta}}(\mathbf{x})]| \\ M_{\mathcal{D}}^{\Delta\text{EO}}(\boldsymbol{\theta}) &= \sum_{y=0}^{1} |\mathbb{E}_{p_{\mathcal{D}}(X=\mathbf{x}|S=0,Y=y)}[h_{\boldsymbol{\theta}}(\mathbf{x})] - \mathbb{E}_{p_{\mathcal{D}}(X=\mathbf{x}|S=1,Y=y)}[h_{\boldsymbol{\theta}}(\mathbf{x})]|, \end{aligned} \tag{6}$$

where $M_a^b(\boldsymbol{\theta})$ denotes the surrogate for the fairness metric b estimated from dataset a. When $\boldsymbol{\theta}$ is itself a function of $\mathbf{w}$, we will use the notation $M_a^b(\boldsymbol{\theta}(\mathbf{w}), \mathbf{w})$.

2.3 Other Related Work

Many pre-processing based bias mitigation algorithms, learn low dimensional representations of the data that are independent of the sensitive attribute [43, 26]. Others aim to learn fairness promoting transformations in the ambient space of the data [7, 32, 40]. Pre-processing methods that transform the data points can often run the risk of losing the semantics of the original data points. Often, they can be expensive, especially for high-dimensional data and large datasets. Furthermore, they must be performed before training any task-specific models and thus are not applicable when the goal is to improve a model already trained with an expensive procedure. In [38], the authors first obtain a counterfactual feature distribution by identifying the test instances, which when dropped the pre-trained model predictions are fair on the remaining test instances. They then learn a optimal transport based randomized pre-processor that maps the transforms the new test samples from the unprivileged group to fair counterfactual distribution. In contrast, our goal is to compute the influence scores for the training instances, which is more challenging. Additionally, [38] requires the sensitive attributes be known at test time as the pre-processor is specific to the unprivileged group. Instead, we aim to directly edit the trained model and eliminate the need of sensitive attribute labels at test time.

In-processing algorithmic fairness methodologies [7, 20] are applicable when we can train models along with fairness constraints. Mary et al. [27] enforce independence through a relaxation of the Hirschfeld-Gebelein-Rényi Maximum Correlation Coefficient (HGR) dependency measure. Similarly, *Rebias* [2, 41] uses the Hilbert-Schmidt Independence Criterion (HSIC) to reduce the dependence of the representations on the sensitive attibutes. *FairMixup* [9] is a data augmentation strategy to improve the generalization properties of in-processing algorithms. These methods can be sensitive to the regularization strength and can sacrifice too much accuracy. In contrast, our approach is applicable when the base model to trained unconstrained on the main task, which can then be updated to remove the influence of the harmful instances and improve fairness.

Existing post-processing methods [19, 17, 30, 25, 39, 41] learn to transform the predictions of a trained model to satisfy a measure of fairness. These can often be limiting as they do not provide control over the fairness accuracy trade-off, may require that predicted scores to be well-calibrated, or may lead to excessive reduction in performance. In contrast, our method exploits the training data and model gradients efficiently to generate stronger, yet computationally inexpensive post-hoc interventions at minimal loss of predictive performance.

3 Fair Classification through Post-Hoc Interventions

We now develop and analyze a post-processing fairness algorithm that given (i) a pre-trained model, (ii) access to the training data and optionally a validation set, (iii) a twice differentiable loss function

[1] Nearly smooth. The absolute value is not differentiable at zero, but this is not a concern since we rarely encounter exact zeros in practice.

and a once differentiable surrogate to the fairness metric, and (iv) an invertible Hessian at a local optimum of the loss, improves the fairness characteristics of a pre-trained model without requiring it to be refit.

3.1 Influence Functions for Group Fairness

Assuming that we use a held-out validation set $\mathcal{D}_{\text{val}} = \{\mathbf{x}_n, s_n, y_n\}_{n=1}^{N_{\text{val}}}$ to estimate $M^{\Delta\text{DP}}_{\mathcal{D}_{\text{val}}}(\boldsymbol{\theta})$ and $M^{\Delta\text{EO}}_{\mathcal{D}_{\text{val}}}(\boldsymbol{\theta})$, we can leverage the result in Equation 5 to compute the influence of the nth training instance on ΔDP,

$$\begin{aligned}\mathcal{I}_{\Delta\text{DP},n} &= -\nabla_{\boldsymbol{\theta}} M^{\Delta\text{DP}}_{\mathcal{D}_{\text{val}}}(\hat{\boldsymbol{\theta}})^T H^{-1} g_n, \\ &= -\nabla_{\theta} \big|\mathbb{E}_{p_{\mathcal{D}}(X=\mathbf{x}|S=1)}[h_{\boldsymbol{\theta}}(\mathbf{x})] - \mathbb{E}_{p_{\mathcal{D}}(X=\mathbf{x}|S=0)}[h_{\boldsymbol{\theta}}(\mathbf{x})]\big|\big|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} H^{-1} g_n,\end{aligned} \tag{7}$$

and on ΔEO,

$$\begin{aligned}\mathcal{I}_{\Delta\text{EO},n} &= -\nabla_{\boldsymbol{\theta}} M^{\Delta\text{EO}}_{\mathcal{D}_{\text{val}}}(\hat{\boldsymbol{\theta}})^T H^{-1} g_n, \\ &= -\nabla_{\theta} \Big|\sum_{y=0}^{1} \mathbb{E}_{p_{\mathcal{D}}(X=\mathbf{x}|S=0,Y=y)}[h_{\boldsymbol{\theta}}(\mathbf{x})] - \mathbb{E}_{p_{\mathcal{D}}(X=\mathbf{x}|S=1,Y=y)}[h_{\boldsymbol{\theta}}(\mathbf{x})]\Big|\Big|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} H^{-1} g_n,\end{aligned} \tag{8}$$

where we have used $M^{\Delta\text{DP}}_{\mathcal{D}_{\text{val}}}(\hat{\boldsymbol{\theta}})$ and $M^{\Delta\text{EO}}_{\mathcal{D}_{\text{val}}}(\hat{\boldsymbol{\theta}})$ to denote $M^{\Delta\text{DP}}_{\mathcal{D}_{\text{val}}}(\boldsymbol{\theta}(\mathbf{w}), \mathbf{w})\big|_{\mathbf{w}=\mathbf{1},\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}}$ and $M^{\Delta\text{EO}}_{\mathcal{D}_{\text{val}}}(\boldsymbol{\theta}(\mathbf{w}), \mathbf{w})\big|_{\mathbf{w}=\mathbf{1},\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}}$. We highlight that computing the influence of training instances on group fairness metrics requires solving a *single* empirical risk minimization problem to recover $\hat{\boldsymbol{\theta}}$. The fairness metrics could also be estimated on the training data if no validation set is available. However, empirically we find that a validation set improves results.

3.2 Post-Hoc Mitigation

Revisiting Equation 3, we note that the first order Taylor approximation about $\mathbf{1}$ is a function of $\mathbf{w}$. This opens up the possibility of post-hoc fairness improvement of a pre-trained $\hat{\boldsymbol{\theta}}$ by searching for a weight vector $\mathbf{w}_{\text{fair}}$ such that $M^b_a(\hat{\boldsymbol{\theta}}_{\text{fair}}) \approx 0$, where,

$$\begin{aligned}\hat{\boldsymbol{\theta}}_{\text{fair}} \stackrel{\text{def}}{=} \hat{\boldsymbol{\theta}}(\mathbf{w}_{\text{fair}}) &= \hat{\boldsymbol{\theta}} + \sum_{n=1}^{N} \frac{d\boldsymbol{\theta}(\mathbf{w})}{dw_n}\bigg|_{\mathbf{w}=\mathbf{1}} (w_n^{\text{fair}} - 1), \\ &= \hat{\boldsymbol{\theta}} - \sum_{n=1}^{N} H^{-1} g_n (w_n^{\text{fair}} - 1),\end{aligned} \tag{9}$$

and $\mathbf{w}_{\text{fair}} = [w_1^{\text{fair}}, w_2^{\text{fair}}, \ldots, w_N^{\text{fair}}]^T \in \mathbb{R}^N$. We could use gradient-based methods to learn $\mathbf{w}_{\text{fair}}$ by optimizing a desired $M(\boldsymbol{\theta}(\mathbf{w}), \mathbf{w})$ with respect to $\mathbf{w}$. However, computing and inverting the Hessian requires $\mathcal{O}(ND^2 + D^3)$ operations and is prohibitively expensive for large models. Instead, iterative procedures involving repeated Hessian-vector products are often used in practice [21]. A gradient-based procedure would need to either perform this iterative procedure after *every* gradient step or pre-compute $\sum_n H^{-1} g_n$, rendering the procedure computationally intractable for most cases of interest. Moreover, solely optimizing $M(\boldsymbol{\theta}(\mathbf{w}), \mathbf{w})$ will likely result in fair but inaccurate classifiers, and the optimized weights will typically not be interpretable.

We circumvent these issues by constraining the elements of $\mathbf{w}$ to be binary. In Proposition 3.1, we show that we can construct $\mathbf{w}_{\text{fair}}$ by simply zeroing out coordinates of $\mathbf{w}_{\text{fair}}$ that correspond to training instances with a positive influence on the fairness metric of interest. This construction is inherently interpretable. Setting an element to zero implies training without the corresponding training instance. Zeroing out instances with positive influence equates to refitting the model after dropping training instances that increase disparity across groups.

We now establish conditions under which $\mathbf{w}_{\text{fair}}$ as constructed above leads to classifiers with lower group disparities. Let $\mathbf{1} \in \mathbb{R}^N$ denote an N-dimensional vector of all ones, b denote a fairness metric, $\overline{M}^b_{\mathcal{D}_{\text{val}}}(\boldsymbol{\theta}(\mathbf{w}), \mathbf{w})$ denote a linearized approximation to $M^b_{\mathcal{D}_{\text{val}}}(\boldsymbol{\theta}(\mathbf{w})), \mathbf{w})$ about $\mathbf{1}$, and $\mathbb{1}[\alpha > \beta]$ denote an indicator function that takes the value one if $\alpha > \beta$ is true and zero otherwise.

Proposition 3.1. *Let* $\mathbf{w}_{\text{fair}} \in \{0,1\}^N$ *be a N dimensional binary vector such that its* n^{th} *coordinate is* $w_n^{\text{fair}} = 1 - \mathbb{1}[\mathcal{I}_{b,n} > 0]$*, then,*

$$\mathbf{w}_{\text{fair}} = \underset{\mathbf{w}\in\{0,1\}^N}{\text{argmin}}\ \overline{M}^b_{\mathcal{D}_{\text{val}}}(\boldsymbol{\theta}(\mathbf{w}), \mathbf{w}) - M^b_{\mathcal{D}_{\text{val}}}(\boldsymbol{\theta}(\mathbf{1}), \mathbf{1}),$$

and $\overline{M}^b_{\mathcal{D}_{\text{val}}}(\boldsymbol{\theta}(\mathbf{w}_{\text{fair}}), \mathbf{w}_{\text{fair}}) - M^b_{\mathcal{D}_{\text{val}}}(\boldsymbol{\theta}(\mathbf{1}), \mathbf{1}) \leq 0.$

Proof. Denote $M^b_{\mathcal{D}_{\text{val}}}(\hat{\boldsymbol{\theta}}) := M_{\mathcal{D}_{\text{val}}}(\boldsymbol{\theta}(\mathbf{1}), \mathbf{1})$. From a first order Taylor approximation about $\mathbf{1}$, we have,

$$\begin{aligned}\overline{M}^b_{\mathcal{D}_{\text{val}}}(\boldsymbol{\theta}(\mathbf{w}), \mathbf{w}) &= M^b_{\mathcal{D}_{\text{val}}}(\hat{\boldsymbol{\theta}}) + \sum_{n=1}^{N} \frac{dM^b_{\mathcal{D}_{\text{val}}}(\hat{\boldsymbol{\theta}}(\mathbf{w}), \mathbf{w})}{dw_n}\Big|_{\mathbf{w}=\mathbf{1}, \boldsymbol{\theta}=\hat{\boldsymbol{\theta}}}(w_n - 1), \\ &= M^b_{\mathcal{D}_{\text{val}}}(\hat{\boldsymbol{\theta}}) + \sum_{n=1}^{N} \mathcal{I}_{b,n}(w_n - 1).\end{aligned} \tag{10}$$

Rearranging terms,

$$\begin{aligned}\overline{M}^b_{\mathcal{D}_{\text{val}}}(\boldsymbol{\theta}(\mathbf{w}), \mathbf{w}) - M^b_{\mathcal{D}_{\text{val}}}(\hat{\boldsymbol{\theta}}) &= \sum_{n=1}^{N} \mathcal{I}_{b,n}(w_n - 1) \\ &= \sum_{n=1}^{N} \mathbb{1}[\mathcal{I}_{b,n} > 0]\mathcal{I}_{b,n}(w_n - 1) + \sum_{n=1}^{N} \mathbb{1}[\mathcal{I}_{b,n} \leq 0]\mathcal{I}_{b,n}(w_n - 1).\end{aligned} \tag{11}$$

Finally, the result follows from observing that $w_n \in \{0, 1\}$ and noting that the first term can be either zero (when $w_n = 1$) or negative (when $w_n = 0$ and $\mathbb{1}[\mathcal{I}_{b,n} > 0]$) and the second term can be either zero (when $w_n = 1$) or positive (when $w_n = 0$ and $\mathbb{1}[\mathcal{I}_{b,n} \leq 0]$). $\mathbf{w}_{\text{fair}}$ drives the second term to zero and sets the first term to the smallest value attainable by a binary $\mathbf{w}$. □

It follows that $M^b_{\mathcal{D}_{\text{val}}}(\boldsymbol{\theta}(\mathbf{w}_{\text{fair}}), \mathbf{w}_{\text{fair}}) \approx\leq M^b_{\mathcal{D}_{\text{val}}}(\hat{\boldsymbol{\theta}}(\mathbf{1}), \mathbf{1})$, with the inequality holding when the linearization is accurate. Finally, defining $\mathcal{D}_- = \{\mathbf{z}_n \mid \mathbf{z}_n \in \mathcal{D} \text{ and } \mathcal{I}_{M,n} > 0\}$, we arrive at the post-hoc mitigated classifier by plugging in $\mathbf{w}_{\text{fair}}$ from Proposition 3.1 in Equation 9,

$$\hat{\boldsymbol{\theta}}_{\text{fair}} = \hat{\boldsymbol{\theta}} + \sum_{m\in\mathcal{D}_-} H^{-1} g_m. \tag{12}$$

In Appendix A we consider an alternate $\mathbf{w}_{\text{fair}}$ that is guaranteed to decrease *both* the loss ℓ and the fairness metric on the validation set $\mathcal{D}_{\text{val}}$. In early experiments, we did not see consistent benefits from using this alternate version and do not consider it further in this paper.

3.3 Practical Considerations

Hessian computation and inversion. The influence function computation involves computing and inverting the Hessian of the loss function on the training data. This requires $\mathcal{O}(ND^2 + D^3)$ operations. Both computing and storing the Hessian becomes prohibitively expensive for large models. While diagonal approximations to the Hessian are possible, they tend to be inaccurate. Instead, iterative methods based on the (truncated) Neumann expansion have been proposed in the past [21]. However, more recent work has found the Neumann approximation to be inaccurate, cf. [36, Appendix C] and prone to numerical issues when the eigenvalues of the Hessian fall outside the $[0, 1]$ interval. Motivated by these shortcomings, here we develop an alternative iterative procedure based on the recently proposed WoodFisher approximation [35].

The WoodFisher approximation provides us with the following recurrence relation for estimating the inverse of the Hessian:

$$\hat{H}_{n+1}^{-1} = \hat{H}_n^{-1} - \frac{\hat{H}_n^{-1}\nabla_\theta \ell(y_{n+1}, h_{\boldsymbol{\theta}}(\mathbf{x}_{n+1}))\nabla_\theta \ell(y_{n+1}, h_{\boldsymbol{\theta}}(\mathbf{x}_{n+1})^\top \hat{H}_n^{-1}}{N + \nabla_\theta \ell(y_{n+1}, h_{\boldsymbol{\theta}}(\mathbf{x}_{n+1}))^\top \hat{H}_n^{-1} \nabla_\theta \ell(y_{n+1}, h_{\boldsymbol{\theta}}(\mathbf{x}_{n+1}))}, \tag{13}$$

with $\hat{H}_0^{-1} = \lambda^{-1} I_D$, and λ a small positive scalar.

For computing influence functions we only need to store the product of the inverse Hessian with a vector $\mathbf{v}$, i.e., $H^{-1}\mathbf{v}$, which should only require $\mathcal{O}(D)$ storage. However, if we first compute the inverse Hessian and then compute the Hessian-vector product (HVP), we would need $\mathcal{O}(D^2)$ storage. To sidestep this issue, we develop the following coupled recurrences that only use $\mathcal{O}(D)$ storage. We call these coupled recurrences *IHVP-WoodFisher*,

$$\boldsymbol{o}_{n+1} = \boldsymbol{o}_n - \frac{\boldsymbol{o}_n \nabla_\theta \ell(\mathbf{z}_{n+1})^\top \boldsymbol{o}_n}{N + \nabla_\theta \ell(\mathbf{z}_{n+1})^\top \boldsymbol{o}_n}, \quad \boldsymbol{k}_{n+1} = \boldsymbol{k}_n - \frac{\boldsymbol{o}_n \nabla_\theta \ell(\mathbf{z}_{n+1})^\top \boldsymbol{k}_n}{N + \nabla_\theta \ell(\mathbf{z}_{n+1})^\top \boldsymbol{o}_n}, \tag{14}$$

where, we use $\ell(\mathbf{z}_{n+1})$ as shorthand for $\ell(y_{n+1}, h_{\boldsymbol{\theta}}(\mathbf{x}_{n+1}))$, $\boldsymbol{o}_1 = \nabla_\theta \ell(y_1, h_{\boldsymbol{\theta}}(\mathbf{x}_1))$, and $\boldsymbol{k}_1 = \mathbf{v}$.

Proposition 3.2. *Let $\boldsymbol{o}_1 = \nabla_\theta \ell(\mathbf{z}_1)$, $\boldsymbol{k}_1 = \mathbf{v}$, and N denote the number of training instances. The Hessian-vector product $H^{-1}\mathbf{v}$ is approximated by iterating through the IHVP-WoodFisher recurrence in Equation* 14 *and computing $\boldsymbol{k}_N$.*

We prove Proposition 3.2 in Appendix B. In practice, we observe that even using $B \ll N$ iterations produces useful approximations. In Appendix C, we compare the approximation accuracy of the the *IHVP-WoodFisher* and the iterative *Neumann* approach on cases where it is tractable to exactly compute the IHVP. Algorithm 2 (see Appendix) summarizes our vanilla approach.

Computational speedups: Although Algorithm 2 suggests running the *IHVP-WoodFisher* iterations for each training instance for clarity of exposition, in practice, we use the following trick to run the *IHVP-WoodFisher* iterations only once for the entire training dataset. First, for any $p * p$ symmetric matrix $\mathbf{A}$ and p-dimensional vectors $\mathbf{x}$ and $\mathbf{y}$, $\mathbf{x}^T\mathbf{A}$, $\mathbf{y} = \mathbf{y}^T\mathbf{A}\mathbf{x}$. From Equation 5, the influence calculation involves computing $\nabla_{\boldsymbol{\theta}} M(\hat{\boldsymbol{\theta}}, \mathbf{1})^T H^{-1} g_n$ for all n in the training dataset. Since is symmetric, we can equivalently compute $g_n^T H^{-1} \nabla_{\boldsymbol{\theta}} M(\hat{\boldsymbol{\theta}}, \mathbf{1})$. We can then run the *IHVP-WoodFisher* iterations to approximate $H^{-1}\nabla_{\boldsymbol{\theta}} M(\hat{\boldsymbol{\theta}}, \mathbf{1})$. Crucially, we need to do this only once. With the approximation in hand, computing the per data influence requires a single dot product per data instance between g_n and the *IHVP-WoodFisher* approximated $H^{-1}\nabla_{\boldsymbol{\theta}} M(\hat{\boldsymbol{\theta}}, \mathbf{1})$. In contrast to other approaches to scaling up influence functions [33] our approach only requires the storage of a single p-dimensional vector. We call this more efficient version `Fair-IJ` and is summarized in Algorithm 1.

Most influential instances. Our development and analysis depends on first order linear approximations of non-linear functions about $\mathbf{1}$. We expect the quality of these approximations to deteriorate further away from $\mathbf{1}$, i.e., with increasing number of instances dropped. See Theorem 1 in [6] for additional discussion on the quality of approximation. We find that instead of dropping all instances with positive influence, dropping the k most influential instances yields better bias mitigation. We select the hyperparameter k that results in the lowest (best) fairness score on the validation set. Additionally, [35] observed that the *WoodFisher* Hessian estimate $\hat{H}$ differs from the true Hessian by a scaling factor, i.e $\hat{H} \propto H$. We select, from a pre-specified set, the scaling factor that minimizes the fairness score on the validation set. We then scale the *IHVP-WoodFisher* estimates using the selected scaling factor. See Appendix D.

4 Experiments

We first study our method on tabular datasets including the well-known Adult dataset [13] and the recently released ACSPublicCoverage [11] dataset. ACSPublicCoverage is one among a suite of datasets aimed to be larger alternatives to previously available fairness datasets. We then investigate our method on the text modality and larger pre-trained models using the CivilComments dataset [5].

4.1 Tabular Datasets

Setup. The task in the Adult dataset is to predict if a person has an income above a threshold. We use gender as the sensitive attribute. This dataset comes with a fixed test set. A random 33% of the training data is used as the validation set for each trial of the experiments. We follow the pre-processing steps from [27]. The task in the ACSPublicCoverage dataset is to predict if a person has public health insurance coverage. For our experiments, we only consider instances from the year 2014, from the state of California, and belonging to the white or black race. We consider race as the

Algorithm 1 `Fair-IJ`

Input: Pre-trained model parameters $\hat{\boldsymbol{\theta}}$, training set $\mathcal{D}$, loss function ℓ, a validation set $\mathcal{D}_{\text{val}}$ and a smooth surrogate to the fairness metric $b \in \{\Delta\text{DP}, \Delta\text{EO}\}$, $M^b_{\mathcal{D}_{\text{val}}}$.
Calculate: $\nabla_{\boldsymbol{\theta}} M(\hat{\boldsymbol{\theta}}, \mathbf{1})$ using Equation 7 or Equation 8.
Calculate: $\boldsymbol{r} = H^{-1} \nabla_{\boldsymbol{\theta}} M(\hat{\boldsymbol{\theta}}, \mathbf{1})$ by setting $\boldsymbol{k}_1 = \nabla_{\boldsymbol{\theta}} M(\hat{\boldsymbol{\theta}}, \mathbf{1})$ and iterating through Equation 14 for B iterations.
Calculate: the fairness influence $\mathcal{I}_{b,n}$ of each training instance $\mathbf{z}_n$ on $\mathcal{D}_{\text{val}}$ by computing dot product between g_n and $\boldsymbol{r}$.
Construct: the set $\mathcal{D}_-$ and denote its cardinality, $|\mathcal{D}_-| = K$.
Initialize: $\hat{\boldsymbol{\theta}}^0_{\text{fair}} := \hat{\boldsymbol{\theta}}$
for $k \in [1, \ldots, K]$ **do**
    Construct: $\mathcal{D}^k_- = \{\mathbf{z}_n \in \mathcal{D}_- \mid \mathcal{I}_{b,n} > \mathcal{I}_{b,(K-k)}\}$, where $\mathcal{I}_{b,(K-k)}$ denotes the $(K-k)^{\text{th}}$ order statistic of the influence scores $[\mathcal{I}_{b,1}, \ldots, \mathcal{I}_{b,K}]$.
    Calculate: $\hat{\boldsymbol{\theta}}^k_{\text{fair}}$ by replacing $\mathcal{D}_-$ with using $\mathcal{D}^k_-$ in Equation 12.
    If $b_{\mathcal{D}_{\text{val}}}(\hat{\boldsymbol{\theta}}^k_{\text{fair}}) < b_{\mathcal{D}_{\text{val}}}(\hat{\boldsymbol{\theta}}^{k-1}_{\text{fair}})$ **set** $\hat{\boldsymbol{\theta}}_{\text{fair}} := \hat{\boldsymbol{\theta}}^k_{\text{fair}}$ **else set** $\hat{\boldsymbol{\theta}}_{\text{fair}} := \hat{\boldsymbol{\theta}}^{k-1}_{\text{fair}}$ and break out of the for loop.
end for
Return: fair model parameters $\hat{\boldsymbol{\theta}}_{\text{fair}}$.

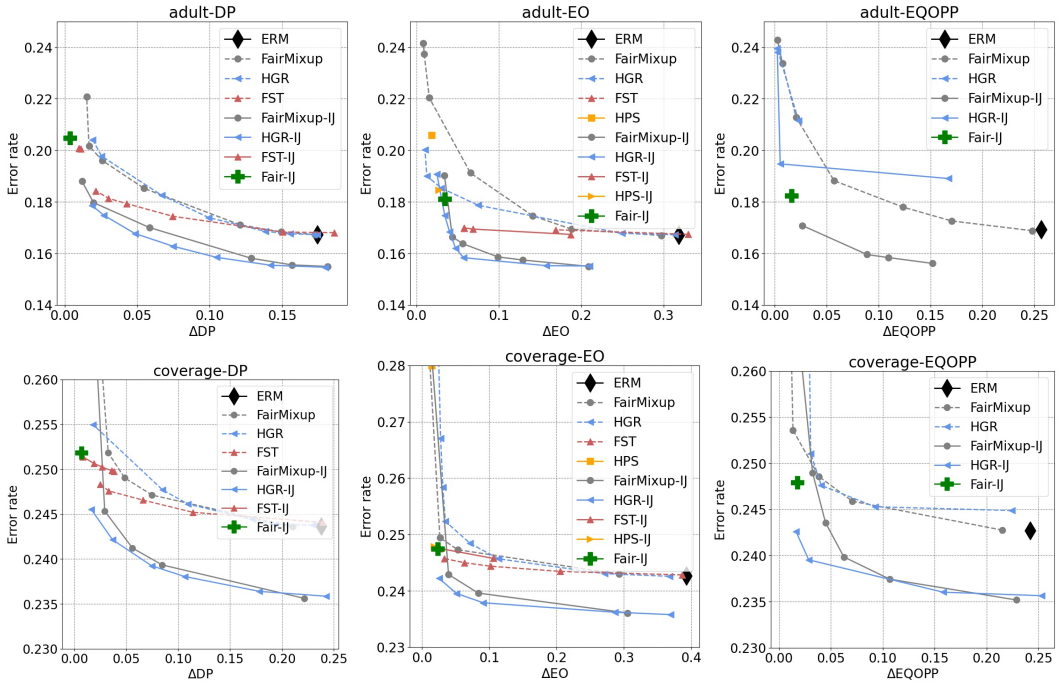

Figure 2: Accuracy and fairness (DP, EO, and EQOPP) Pareto frontier for the Adult and the Coverage datasets averaged over 10 runs. Points closer to the bottom-left achieve the best fairness/accuracy trade-off.

sensitive attribute. In the rest of the paper, we refer to this subset simply as *Coverage* dataset. We randomly split the dataset into train/validation/test with partition 50% / 20% / 30%, respectively, for each trial. Additionally, for both datasets we standardize the features before training our methods and the baselines.

We first train a 1-hidden layer fully connected neural network with *SeLu* activation function and 100 hidden units. This initial model is trained using standard ERM loss with batch size set to 256. We use the Adam optimizer with learning rate set to 10^{-4}. We train the ERM and baselines for 100 epochs and pick the checkpoint with best accuracy on the validation set. We then employ Algorithm 1 to

arrive at the Fair-IJ solution. We select k and the IHVP scaling term based on the same validation set used to compute the influence scores.

Compared algorithms. We compare our method to several in-processing and post-processing bias mitigation algorithms that are applicable to a wide range of model classes including deep neural networks. We omit the comparisons with pre-processing methods as our goal is to improve a given pre-trained model. Among in-processing algorithms, we compare with: *FairMixup* [9] and *HGR* [27]. *FairMixup* achieves fairness through a *mixup* based regularization employed during training. The *HGR* approach proposes a surrogate to HGR dependence measure and promotes fairness during training by enforcing conditional independence implied by the fairness metrics. On both of these methods, the fairness accuracy trade-off is achieved through the strength of the regularizer.

Among post-processing methods, we compare with *FST* [39] and *HPS* [17]. *FST* optimally transforms the pre-trained model's prediction scores to satisfy a specified fairness constraint and supports DP and EO metrics. To improve the performance of *FST* we re-calibrate the prediction scores using isotonic Regression. *HPS* is designed to enforce EO and requires knowledge of the sensitive attribute at test time. We use a fixed pre-trained model, architecture and training procedure for all the baselines. We also run the baselines on the edited model obtained from the output of our `Fair-IJ` algorithm, $\hat{\boldsymbol{\theta}}_{\text{fair}}$. Specifically, we fine-tune the `Fair-IJ` solution using the in-processing algorithms. In the case of post-processing algorithms, we directly apply these to the `Fair-IJ` edited model.

Results. Figure 2 shows the accuracy and fairness Pareto frontier for the Adult and the Coverage datasets averaged over 10 runs. It can be seen that `Fair-IJ` consistently produces lower disparities across datasets and metrics. Moreover, we observe that baselines operating on $\hat{\boldsymbol{\theta}}_{\text{fair}}$, FairMixup-IJ, HGR-IJ, FST-IJ, and HPS-IJ often achieve substantially better accuracy/fairness trade-off over their counterparts. In Figure 1, we plot the sorted influence scores of training instances on the average demographic parity of a held-out validation set for different datasets considered in this paper.

4.2 CivilComments Dataset

Setup. The *CivilComments* dataset [5] consists of human-annotated attributes on hate comments posted on the internet. The task here is to predict whether a particular comment is toxic. Prior work has shown that automatic toxicity classifiers can achieve sub-optimal performance on certain subpopulations [28, 12, 31]. The goal is to apply our approach to mitigate bias in pre-trained toxicity classifiers. In our experiments we consider `Muslim` as the sensitive attribute. Similar to [22], we assign an instance to the unprivileged group whenever it is annotated with that attribute and assign the rest to the privileged group.

To show the adaptability of our method on large neural networks, we consider three different variants of the pre-trained frozen BERT [10], where features are augmented with: a) $\texttt{BERT}_{\texttt{LC}}$: a linear classifier head, b) $\texttt{BERT}_{\texttt{NC}}$: a non-linear classifier head and c) $\texttt{BERT}_{\texttt{TT=n}}$: with n trigger-tokens in the embedding layer. The last variant is an extension of prompt-tuning [23] or prefix-tuning [24] methods, which are more powerful ways of fine-tuning large-language models than only updating classifier heads. It is worth noting that the adaptation of trigger-tokens scale fittingly in optimizing weights in Equation 9. We compare our results to the simple yet effective method of Gap Regularization (GapReg) from [9] where a model optimization is regularized by a fairness measure added to the training loss while scaled by λ factor to control the regularizer magnitude, as defined in Equation 1 of [9].

Results. We present our results in Table 1. In comparison to the baseline methods ERM and GapReg, `Fair-IJ` consistently performs better in mitigating the group disparities. Additionally, `Fair-IJ` manages to have a better task performance (balanced accuracy for toxicity classification) trade-off while attempting to achieve a lower disparity. In Table 1, we also present the results on virtual trigger-tokens, which we notice to be performing equally well in lowering the disparity. This is a significant observation as it shows how `Fair-IJ` can be efficiently integrated with large neural network through the scalable influence calculations of relatively few trigger parameters. Further training details, observations and baselines are presented in the Appendix.

5 Conclusion

In this work, we proposed `Fair-IJ`, an infinitesimal jackknife-based approach to mitigate the influence of biased training data points without refitting the model. Our approach is limited to

Model	BA	ΔEO	BA	ΔDP
$\texttt{BERT}_{\texttt{LC}}$-ERM	57.3	0.314	58.4	0.246
$\texttt{BERT}_{\texttt{LC}}$-GapReg	57.9	0.133	57.5	0.185
$\texttt{BERT}_{\texttt{LC}}$-Fair-IJ	58.6	0.125	57.1	0.011
$\texttt{BERT}_{\texttt{NC}}$-ERM	59.3	0.326	59.3	0.261
$\texttt{BERT}_{\texttt{NC}}$-GapReg	59.1	0.144	58.2	0.054
$\texttt{BERT}_{\texttt{NC}}$-Fair-IJ	59.8	0.126	58.6	0.008

Model	BA	ΔEO	BA	ΔDP
$\texttt{BERT}_{\texttt{TT=4}}$-ERM	57.5	0.360	57.5	0.268
$\texttt{BERT}_{\texttt{TT=4}}$-Fair-IJ	56.0	0.102	55.2	0.042
$\texttt{BERT}_{\texttt{TT=8}}$-ERM	58.2	0.317	58.2	0.254
$\texttt{BERT}_{\texttt{TT=8}}$-Fair-IJ	56.5	0.113	56.4	0.071
$\texttt{BERT}_{\texttt{TT=10}}$-ERM	58.3	0.348	58.3	0.234
$\texttt{BERT}_{\texttt{TT=10}}$-Fair-IJ	57.2	0.110	56.6	0.089

Table 1: Comparison between ERM, Gap Regularization (for $\lambda = 1$), and Fair-IJ for CivilComments on the sensitive attribute MUSLIM when we use pre-trained BERT model. We report the difference in equality of odds (ΔEO), difference in demographic parity (ΔDP), along with the task balanced accuracy (BA).

settings where the assumptions listed in Section 3 hold. Also, care must be taken to choose an appropriate fairness criterion and its differentiable surrogate for any given application to avoid unwanted consequences. We restricted our analysis to binary classification, since this is by far the most common setup in the fairness literature, but our approach applies to any metric M that is once differentiable in the model parameters and any training loss that is twice differentiable in the parameters. This includes standard approaches to multiclass classification and regression.

Future work includes extending our approach to black-box models where the gradients are inaccessible and incorporating higher-order Taylor approximations to improve the accuracy of the influence functions. We hope that our method further encourages researchers and practitioners in studying and applying bias mitigation to diverse and complex models and datasets.

References

[1] Naman Agarwal, Brian Bullins, and Elad Hazan. Second-order stochastic optimization for machine learning in linear time. *Journal of Machine Learning Research*, 18(1):4148–4187, 2017.

[2] Hyojin Bahng, Sanghyuk Chun, Sangdoo Yun, Jaegul Choo, and Seong Joon Oh. Learning de-biased representations with biased representations. In *International Conference on Machine Learning*, pages 528–539, 2020.

[3] Solon Barocas, Moritz Hardt, and Arvind Narayanan. *Fairness and Machine Learning*. fairmlbook.org, 2019. http://www.fairmlbook.org.

[4] Samyadeep Basu, Philip Pope, and Soheil Feizi. Influence functions in deep learning are fragile. *arXiv preprint arXiv:2006.14651*, 2020.

[5] Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. Nuanced metrics for measuring unintended bias with real data for text classification. In *Companion Proceedings of the World Wide Web Conference*, pages 491–500, 2019.

[6] Tamara Broderick, Ryan Giordano, and Rachael Meager. An automatic finite-sample robustness metric: When can dropping a little data make a big difference? *arXiv preprint arXiv:2011.14999*, 2020.

[7] Flavio Calmon, Dennis Wei, Bhanukiran Vinzamuri, Karthikeyan Natesan Ramamurthy, and Kush R. Varshney. Optimized pre-processing for discrimination prevention. *Advances in Neural Information Processing Systems*, 30, 2017.

[8] Ran Canetti, Aloni Cohen, Nishanth Dikkala, Govind Ramnarayan, Sarah Scheffler, and Adam Smith. From soft classifiers to hard decisions: How fair can we be? In *Conference on Fairness, Accountability, and Transparency*, pages 309–318, 2019.

[9] Ching-Yao Chuang and Youssef Mroueh. Fair mixup: Fairness via interpolation. *arXiv preprint arXiv:2103.06503*, 2021.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[11] Frances Ding, Moritz Hardt, John Miller, and Ludwig Schmidt. Retiring adult: New datasets for fair machine learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[12] Lucas Dixon, John Li, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. Measuring and mitigating unintended bias in text classification. In *AAAI/ACM Conference on AI, Ethics, and Society*, pages 67–73, 2018.

[13] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[14] Bradley Efron. Nonparametric estimates of standard error: The jackknife, the bootstrap and other methods. *Biometrika*, 68(3):589–599, 1981.

[15] Soumya Ghosh, Will Stephenson, Tin D. Nguyen, Sameer Deshpande, and Tamara Broderick. Approximate cross-validation for structured models. *Advances in Neural Information Processing Systems*, 33:8741–8752, 2020.

[16] Ryan Giordano, Michael I. Jordan, and Tamara Broderick. A higher-order Swiss Army infinitesimal jackknife. *arXiv preprint arXiv:1907.12116*, 2019.

[17] Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. In *Advances in Neural Information Processing Systems*, pages 3315–3323, 2016.

[18] L. Jaeckel. The infinitesimal jackknife, memorandum. Technical report, MM 72-1215-11, Bell Lab. Murray Hill, NJ, 1972.

[19] Faisal Kamiran, Asim Karim, and Xiangliang Zhang. Decision theory for discrimination-aware classification. In *IEEE International Conference on Data Mining*, pages 924–929, 2012.

[20] Toshihiro Kamishima, Shotaro Akaho, and Jun Sakuma. Fairness-aware learning through regularization approach. In *IEEE International Conference on Data Mining Workshops*, pages 643–650, 2011.

[21] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, pages 1885–1894, 2017.

[22] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony Lee, Etienne David, Ian Stavness, Wei Guo, Berton A. Earnshaw, Imran S. Haque, Sara Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. WILDS: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, 2021.

[23] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

[24] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

[25] Pranay K. Lohia, Karthikeyan Natesan Ramamurthy, Manish Bhide, Diptikalyan Saha, Kush R. Varshney, and Ruchir Puri. Bias mitigation post-processing for individual and group fairness. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2847–2851, 2019.

[26] Christos Louizos, Kevin Swersky, Yujia Li, Max Welling, and Richard Zemel. The variational fair autoencoder. *arXiv preprint arXiv:1511.00830*, 2015.

[27] Jérémie Mary, Clément Calauzenes, and Noureddine El Karoui. Fairness-aware learning for continuous attributes and treatments. In *International Conference on Machine Learning*, pages 4382–4391, 2019.

[28] Ji Ho Park, Jamin Shin, and Pascale Fung. Reducing gender bias in abusive language detection. *arXiv preprint arXiv:1808.07231*, 2018.

[29] Pouya Pezeshkpour, Sarthak Jain, Byron C. Wallace, and Sameer Singh. An empirical comparison of instance attribution methods for NLP. *arXiv preprint arXiv:2104.04128*, 2021.

[30] Geoff Pleiss, Manish Raghavan, Felix Wu, Jon Kleinberg, and Kilian Q. Weinberger. On fairness and calibration. *Advances in Neural Information Processing Systems*, 30, 2017.

[31] Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. Distributionally robust neural networks. In *International Conference on Learning Representations*, 2019.

[32] Prasanna Sattigeri, Samuel C. Hoffman, Vijil Chenthamarakshan, and Kush R. Varshney. Fairness GAN. *arXiv preprint arXiv:1805.09910*, 2018.

[33] Andrea Schioppa, Polina Zablotskaia, David Vilar, and Artem Sokolov. Scaling up influence functions. In *AAAI Conference on Artificial Intelligence*, volume 36, pages 8179–8186, 2022.

[34] Jonathan Richard Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical report, Carnegie Mellon University, 1994.

[35] Sidak Pal Singh and Dan Alistarh. Woodfisher: Efficient second-order approximation for neural network compression. *Advances in Neural Information Processing Systems*, 33:18098–18109, 2020.

[36] William T. Stephenson and Tamara Broderick. Approximate cross-validation in high dimensions with guarantees. In *International Conference on Artificial Intelligence and Statistics*, 2020.

[37] Kush R. Varshney. *Trustworthy Machine Learning*. Independently Published, Chappaqua, NY, USA, 2022.

[38] Hao Wang, Berk Ustun, and Flavio Calmon. Repairing without retraining: Avoiding disparate impact with counterfactual distributions. In *International Conference on Machine Learning*, pages 6618–6627, 2019.

[39] Dennis Wei, Karthikeyan Natesan Ramamurthy, and Flavio Calmon. Optimized score transformation for fair classification. In *International Conference on Machine Learning*, pages 1673–1683, 2020.

[40] Depeng Xu, Shuhan Yuan, Lu Zhang, and Xintao Wu. FairGAN: Fairness-aware generative adversarial networks. In *IEEE International Conference on Big Data*, pages 570–575, 2018.

[41] Yilun Xu, Hao He, Tianxiao Shen, and Tommi S. Jaakkola. Controlling directions orthogonal to a classifier. In *International Conference on Learning Representations*, 2021.

[42] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P. Gummadi. Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment. In *International Conference on World Wide Web*, pages 1171–1180, 2017.

[43] Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. Learning fair representations. In *International Conference on Machine Learning*, pages 325–333, 2013.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See the Conclusion section.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See the Conclusion section.

(d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See the Section 3.
   (b) Did you include complete proofs of all theoretical results? [Yes] See the Sections 3.2, 3.3 and the Appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] The code is proprietary.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See the Appendix.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See the Appendix.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See the Appendix.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] See the Appendix.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix

A Loss aware $\mathbf{w}_{\text{fair}}$

While this construction of $\mathbf{w}_{\text{fair}}$ reduces the fairness metric it pays no heed to the original loss, ℓ, and may lead to classifiers that are less accurate than $\hat{\boldsymbol{\theta}}$. To account for ℓ we additionally compute $\mathcal{I}_{\ell,n}$ the influence of each training instance on the validation loss, $L_{\mathcal{D}_{\text{val}}}(\boldsymbol{\theta}) = \frac{1}{N_{\text{val}}} \sum_{n=1}^{N_{\text{val}}} \ell(y_n, h_{\boldsymbol{\theta}}(\mathbf{x}_n))$ and set,

$$w_n^{\text{fair}} = 1 - \mathbb{1}[\mathcal{I}_{M,n} > 0]\mathbb{1}[\mathcal{I}_{\ell,n} > 0], \tag{15}$$

i.e., we zero out those coordinates of $\mathbf{w}_{\text{fair}}$ that correspond to training instances with a positive influence on both the fairness metric and the loss ℓ. Finally, defining $\mathcal{D}_- = \{\mathbf{z}_n \mid \mathbf{z}_n \in \mathcal{D} \text{ and } \mathcal{I}_{M,n} > 0 \ \& \ \mathcal{I}_{\ell,n} > 0\}$, we arrive at the post-hoc mitigated classifier by plugging in the zeroed-out $\mathbf{w}_{\text{fair}}$ in Equation 9,

$$\hat{\boldsymbol{\theta}}_{\text{fair}} = \hat{\boldsymbol{\theta}} + \sum_{m \in \mathcal{D}_-} H^{-1} g_m. \tag{16}$$

B IHVP-WoodFisher

In this section, we show that the coupled recurrences, that we refer to as *IHVP-WoodFisher*, computes the Inverse-Hessian Vector Product (IHVP). We begin by restating Proposition 3.2,

Proposition B.1. *Let $\boldsymbol{o}_1 = \nabla_\theta \ell(\mathbf{z}_1)$, $\boldsymbol{k}_1 = \mathbf{v}$, and N denote the number of training instances. The Hessian-vector product $H^{-1}\mathbf{v}$ is approximated by iterating through the IHVP-WoodFisher recurrence in Equation* 14 *and computing $\boldsymbol{k}_N$.*

Proof. The WoodFisher approximation provides us with the following recurrence relation for estimating the inverse of the Hessian,

$$H_{n+1}^{-1} = H_n^{-1} - \frac{H_n^{-1}\nabla_\theta \ell(\mathbf{z}_{n+1})\nabla_\theta \ell(\mathbf{z}_{n+1})^\top H_n^{-1}}{N + \nabla_\theta \ell(\mathbf{z}_{n+1})^\top H_n^{-1} \nabla_\theta \ell(\mathbf{z}_{n+1})} \tag{17}$$

with, $H_0^{-1} = \lambda^{-1} I_D$, and λ is small positive scalar.

Multiplying, both sides by $\nabla_\theta \ell(\mathbf{z}_{n+1})$, we get,

$$H_{n+1}^{-1}\nabla_\theta \ell(\mathbf{z}_{n+1}) = H_n^{-1}\nabla_\theta \ell(\mathbf{z}_{n+1}) - \frac{H_n^{-1}\nabla_\theta \ell(\mathbf{z}_{n+1})\nabla_\theta \ell(\mathbf{z}_{n+1})^\top H_n^{-1}\nabla_\theta \ell(\mathbf{z}_{n+1})}{N + \nabla_\theta \ell(\mathbf{z}_{n+1})^\top H_n^{-1} \nabla_\theta \ell(\mathbf{z}_{n+1})} \tag{18}$$

By substituting $H_n^{-1}\nabla_\theta \ell(\mathbf{z}_{n+1})$ with $\boldsymbol{o}_n$ and assuming $\nabla_\theta \ell(\mathbf{z}_{n+1})$ and $\nabla_\theta \ell(\mathbf{z}_{n+2})$ are close, we construct the following recurrence relation

$$\boldsymbol{o}_{n+1} = \boldsymbol{o}_n - \frac{\boldsymbol{o}_n \nabla_\theta \ell(\mathbf{z}_{n+1})^\top \boldsymbol{o}_n}{N + \nabla_\theta \ell(\mathbf{z}_{n+1})^\top \boldsymbol{o}_n} \tag{19}$$

Now, multiplying both sides of Equation 17 by $\mathbf{v}$, gives us the recurrence relation for the IHVP,

$$H_{n+1}^{-1}\mathbf{v} = H_n^{-1}\mathbf{v} - \frac{H_n^{-1}\nabla_\theta \ell(\mathbf{z}_{n+1})\nabla_\theta \ell(\mathbf{z}_{n+1})^\top H_n^{-1}\mathbf{v}}{N + \nabla_\theta \ell(\mathbf{z}_{n+1})^\top H_n^{-1} \nabla_\theta \ell(\mathbf{z}_{n+1})} \tag{20}$$

By substituting $H_{n+1}^{-1}\mathbf{v}$ with $\boldsymbol{k}_{n+1}$ and $H_n^{-1}\nabla_\theta \ell(\mathbf{z}_{n+1})$ with $\boldsymbol{o}_n$, we get

$$\boldsymbol{k}_{n+1} = \boldsymbol{k}_n - \frac{\boldsymbol{o}_n \nabla_\theta \ell(\mathbf{z}_{n+1})^\top \boldsymbol{k}_n}{N + \nabla_\theta \ell(\mathbf{z}_{n+1})^\top \boldsymbol{o}_n} \tag{21}$$

Thus, under our assumptions, when H_n^{-1} converges to H^{-1}, $\boldsymbol{k}_n$ converges to the IHVP $H^{-1}\mathbf{v}$.

□

C IHVP-WoodFisher Approximation Accuracy

In this section, we discuss the approximation accuracy of *IHVP-WoodFisher* and compare it with *IHVP-Neumann*.

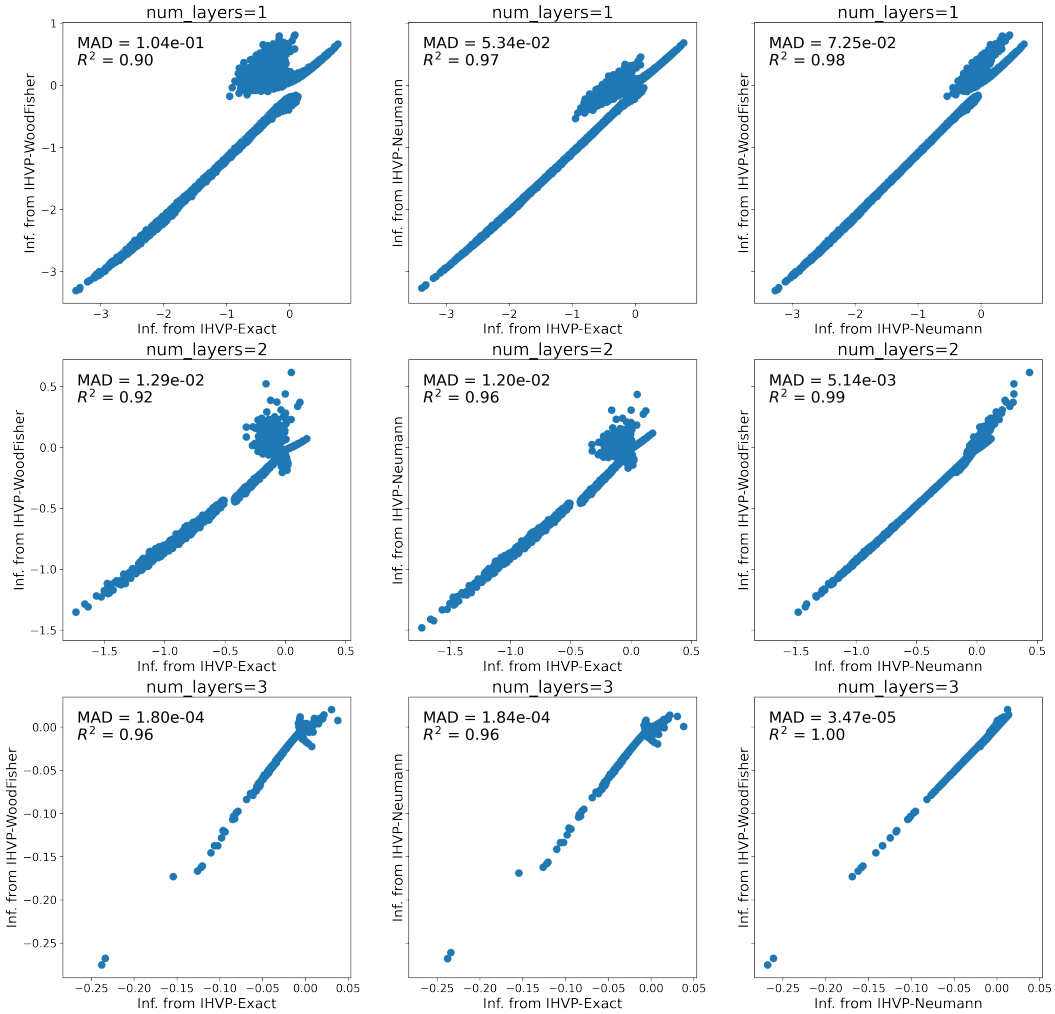

Figure 3: **IHVP-WoodFisher Approximation Accuracy.** The plots show the accuracy of the *IHVP-WoodFisher* and *IHVP-Neumann* approximations by comparing with influence scores obtained from *IHVP-Exact*. We train models with different number of layers on the linearly separable two moon dataset and report Median Absolute Deviation (MAD) and R^2 scores.

IHVP-WoodFisher relies on the assumption that the empirical Fisher matrix is a good approximation to the Hessian of the loss. The Hessian of the loss is known to converge to the true Fisher matrix, when: a) the loss of the model used during training can be expressed as negative log likelihood; and b) the model likelihood has converged to the true data likelihood [35]. The empirical Fisher matrix does not have convergence guarantees as the true Fisher but, it is computationally cheap and works well in practice as an approximation to the Hessian matrix. This is also seen in our experimental results.

To study the approximation accuracy of *IHVP-WoodFisher* and compare with the *Exact* approach of computing IHVP, we consider a setting where the number of parameters is small; specifically, we generate a linearly separable variant of the two moon dataset consisting of 10000 points, where each point has 2 input features and can belong to one of the two classes. We create at $80 - 20$ train-test split and train models with depth 1, 2, and 3 to observe the effect of depth. Hidden layers have a fixed width of 5 units. We use Adam optimizer with learning rate 0.001. After training, we pick

Algorithm 2 `Fair-IJ (slow)`

Input: Pre-trained model parameters $\hat{\boldsymbol{\theta}}$, training set $\mathcal{D}$, loss function ℓ, a validation set $\mathcal{D}_{\text{val}}$ and a smooth surrogate to the fairness metric $b \in \{\Delta\text{DP}, \Delta\text{EO}\}$, $M^b_{\mathcal{D}_{\text{val}}}$.
Calculate: $H^{-1}g_n$ for each training instance $\mathbf{z}_n$ by setting $\boldsymbol{k}_1 = g_n$ and iterating through Equation 14 for B iterations.
Calculate: the fairness influence $\mathcal{I}_{b,n}$ of each training instance $\mathbf{z}_n$ on $\mathcal{D}_{\text{val}}$ using Equation 7 or Equation 8.
Construct: the set $\mathcal{D}_-$ and denote its cardinality, $|\mathcal{D}_-| = K$.
Initialize: $\hat{\boldsymbol{\theta}}^0_{\text{fair}} := \hat{\boldsymbol{\theta}}$
for $k \in [1, \ldots, K]$ **do**
    Construct: $\mathcal{D}^k_- = \{\mathbf{z}_n \in \mathcal{D}_- \mid \mathcal{I}_{b,n} > \mathcal{I}_{b,(K-k)}\}$, where $\mathcal{I}_{b,(K-k)}$ denotes the $(K-k)^{\text{th}}$ order statistic of the influence scores $[\mathcal{I}_{b,1}, \ldots, \mathcal{I}_{b,K}]$.
    Calculate: $\hat{\boldsymbol{\theta}}^k_{\text{fair}}$ by replacing $\mathcal{D}_-$ with using $\mathcal{D}^k_-$ in Equation 12.
    If $b_{\mathcal{D}_{\text{val}}}(\hat{\boldsymbol{\theta}}^k_{\text{fair}}) < b_{\mathcal{D}_{\text{val}}}(\hat{\boldsymbol{\theta}}^{k-1}_{\text{fair}})$ **set** $\hat{\boldsymbol{\theta}}_{\text{fair}} := \hat{\boldsymbol{\theta}}^k_{\text{fair}}$ **else set** $\hat{\boldsymbol{\theta}}_{\text{fair}} := \hat{\boldsymbol{\theta}}^{k-1}_{\text{fair}}$ and break out of the for loop.
end for
Return: fair model parameters $\hat{\boldsymbol{\theta}}_{\text{fair}}$.

a random point from the test set and compute the influence score of the training instances using *IHVP-WoodFisher* and *IHVP-Neumann* approximations as well as exactly computing the IHVPs. For both approximations we use 1000 iterations and average over 10 runs. The *IHVP-Neumann* approximation has an additional hyper-parameter - *scale*. This is to ensure that the Eigenvalues of the Hessian are between $[0, 1]$. For *IHVP-Neumann*'s convergence, *scale* has to be greater than the largest Eigenvalue of the Hessian. In these experiments, we set this hyperparameter to 25.0 which is larger than the largest Eigenvalue we observed for all the models we trained.

In Figure 3 we compare the influence scores and report the Median Absolute Deviation (MAD) and R^2 scores. For each depth value, when plotting and computing the metrics we rescale the influence scores from both approximations to match the mean of the *IHVP-Exact*. It can be observed that both *IHVP-Neumann* and *IHVP-WoodFisher* approaches match well with the influence scores obtained from *Exact* computation. The main benefit of *IHVP-WoodFisher* is that it does not require higher-order gradients. Additionally, unlike *IHVP-Neumann*, this method does not require expensive hyperparameter search to rescale the Eigenvalues of Hessian to ensure IHVP computation convergence.

D Additional Dataset, Training Details and Results

In Table 2, we provide additional information information about the three datasets – Adult[2], Coverage [3], and CivilComments[4]. We trained our models on NVIDIA A100 Tensor Core GPUs. In the case of the tabular datasets, 10 runs with a particular fairness metric took less than 2 hours. This includes training the base model using ERM followed by the application of our `Fair-IJ` algorithm. Within the algorithm, we search for the best k among 40 values spread uniformly in the range in the ranges $0 - 2000$. Similarly, for the IHVP scaling we select the best value among $(0.01, 0.1, 1.0, 2.0, 3.0, 5.0, 10.0)$. This search only requires inference over the validation and hence is relatively inexpensive. The post-processing baselines (*FST* and *HPS*), assume access to a pre-trained model. Similar to our approach they use the validation data to mitigate bias in the pre-trained models. For the in-processing baselines (*HGR* and *FairMixup*), following standard practice, we train the models on the training set and use the validation set to select the hyper-parameter that determines the strength of the fairness regularizer employed by these methods. In Figure 4, we reproduce the Figure 2 with error bars for both the accuracy and fairness metrics.

We now provide additional details regarding the experiments on CivilComments datasets. Table 3 presents additional results comparing ERM models (built without any fairness adjustment) to models regularized using Gap Regularization (GapReg) from *FairMixup* [9], Hirschfeld-Gebelein-Rényi

[2] `https://archive.ics.uci.edu/ml/datasets/adult`
[3] `https://github.com/zykls/folktables`
[4] `https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification/data`

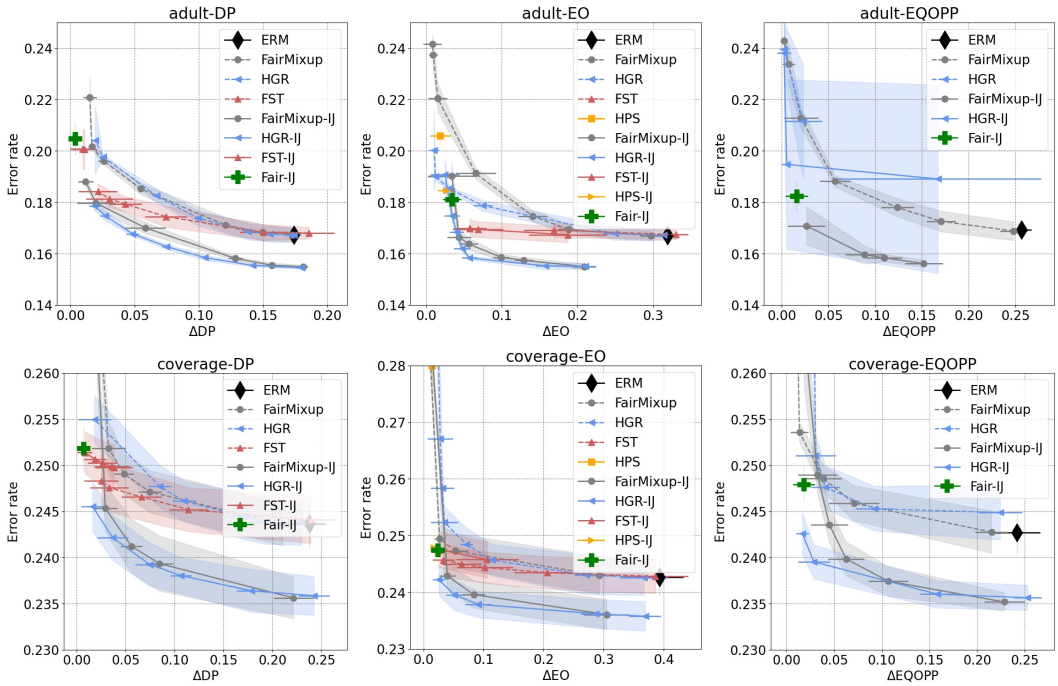

Figure 4: Accuracy and fairness Pareto frontier with error bars for the Adult and the Coverage datasets averaged over 10 runs. Points closer to the bottom-left achieve the best fairness/accuracy trade-off.

Table 2: Summary of datasets.

Dataset	**Target**	**Attribute**	**Size** Train / Valid / Test
Adult	Income	Sex	21815 / 10746 / 12661
Coverage	Health Insurance Coverage	Race	44168 / 21755 / 32471
Civil Comments	Toxicity	Muslim	269038 / 45180 / 133782

Maximum Correlation Coefficient (HGR) dependency measure [27], and *Fair-IJ*. Reported results are for the sensitive attribute `MUSLIM`. They include Equality of odds (ΔEO), demographic parity (ΔDP), and balanced accuracy (BA) and are the means and standard deviations over 5 training runs, each with a different random seed (i.e. 5 different seeds for each configuration). Each model was built with 100 epochs of SGD over the training data for a total of 24h of computation time (using a NVIDIA A100 GPU). All reported results were computed on the test dataset for models with the best validation loss over the 100 epochs of training (models being validated at the end of each epoch). For loss-regularized models using GapReg and HGR, values of $\lambda = 1.0$ and $\lambda = 0.2$ were used to add the regularizer term to the training loss; both sets of results are given in Table 3. Inference on the test dataset is quite fast (3 minutes) on NVIDIA A100 GPUs. Results for both $\texttt{BERT}_{\texttt{LC}}$ (Linear Classifier) and $\texttt{BERT}_{\texttt{NC}}$ (Non-linear Classifier) are provided for ERM, GapReg, HGR, and `Fair-IJ`.

Both our $\texttt{BERT}_{\texttt{LC}}$ and $\texttt{BERT}_{\texttt{NC}}$ architectures are defined based on different classification layer(s) on top of a pre-trained BERT model. In all our experiments, we use $\text{BERT}_{\text{base}}$ model. While $\texttt{BERT}_{\texttt{LC}}$ uses just a single dense layer on top of the pooled vector from BERT representations, $\texttt{BERT}_{\texttt{NC}}$ has multiple dense layers on top of the pooled output. For the latter, we use two dense layers with hidden sizes of 768 and 128 intertwined with ReLU non-linearities. As described in 4.2, we also provide quantitative results on a variant of $\texttt{BERT}_{\texttt{LC}}$, referred as $\texttt{BERT}_{\texttt{TT=N}}$, which uses virtual tokens, where `N` refers to number of trigger tokens. In this setup, we introduce a new parameterized embedding, called *trigger* embeddings, which are learned during the training. Similar to methods introduced in [23] and [24], we add trigger tokens to each sequence during the training. In our quantitative analysis we use

Model	BA	ΔEO	BA	ΔDP
$\texttt{BERT}_{\texttt{LC}}$-ERM	58.4±0.229	0.314±0.026	58.4±0.229	0.246±0.011
$\texttt{BERT}_{\texttt{NC}}$-ERM	59.3±0.025	0.326±0.006	59.3±0.025	0.261±0.003
$\texttt{BERT}_{\texttt{LC}}$-GapReg ($\lambda=0.2$)	58.3±0.000	0.198±0.006	58.3±0.000	0.149±0.004
$\texttt{BERT}_{\texttt{LC}}$-GapReg ($\lambda=1.0$)	57.9±0.001	0.144±0.010	56.9±0.016	0.071±0.009
$\texttt{BERT}_{\texttt{NC}}$-GapReg ($\lambda=0.2$)	59.0±0.000	0.166±0.007	58.8±0.005	0.136±0.010
$\texttt{BERT}_{\texttt{NC}}$-GapReg ($\lambda=1.0$)	57.9±0.011	0.144±0.067	58.2±0.003	0.054±0.028
$\texttt{BERT}_{\texttt{LC}}$-HGR ($\lambda=0.2$)	56.9±0.018	0.209±0.075	58.3±0.000	0.213±0.002
$\texttt{BERT}_{\texttt{LC}}$-HGR ($\lambda=1.0$)	53.0±0.002	0.372±0.015	58.1±0.000	0.149±0.003
$\texttt{BERT}_{\texttt{NC}}$-HGR ($\lambda=0.2$)	59.0±0.000	0.169±0.007	59.1±0.003	0.240±0.004
$\texttt{BERT}_{\texttt{NC}}$-HGR ($\lambda=1.0$)	54.0±0.003	0.339±0.042	59.1±0.000	0.191±0.002
$\texttt{BERT}_{\texttt{LC}}$-Fair-IJ	58.6±0.162	0.125±0.011	57.1±0.237	0.011±0.004
$\texttt{BERT}_{\texttt{NC}}$-Fair-IJ	59.8±0.326	0.126±0.011	58.6±0.170	0.008±0.004

Model	BA	ΔEO	BA	ΔDP
$\texttt{BERT}_{\texttt{TT=4}}$-ERM	57.5±0.794	0.360±0.090	57.5±0.794	0.268±0.037
$\texttt{BERT}_{\texttt{TT=8}}$-ERM	58.2±0.179	0.317±0.060	58.2±0.179	0.254±0.032
$\texttt{BERT}_{\texttt{TT=10}}$-ERM	58.3±0.842	0.348±0.070	58.3±0.842	0.234±0.043
$\texttt{BERT}_{\texttt{TT=4}}$-Fair-IJ	56.0±0.922	0.102±0.035	55.2±1.202	0.042±0.060
$\texttt{BERT}_{\texttt{TT=8}}$-Fair-IJ	56.5±0.690	0.113±0.040	56.4±0.512	0.071±0.069
$\texttt{BERT}_{\texttt{TT=10}}$-Fair-IJ	57.2±1.483	0.110±0.045	56.6±1.349	0.089±0.056

Table 3: Comparison between ERM, Gap Regularization, Hirschfeld-Gebelein-Renyi Maximum Correlation Coefficient (HGR) (both for $\lambda = 0.2, 1.0$), and `Fair-IJ` for CivilComments on the sensitive attribute `MUSLIM`. We report the mean and standard deviation of difference in equality of odds (ΔEO), difference in demographic parity (ΔDP), along with the task balanced accuracy (BA) on 5 different seeds.

variants with 4, 8, and 10 trigger tokens which are referred as $\texttt{BERT}_{\texttt{TT=4}}$, $\texttt{BERT}_{\texttt{TT=8}}$, and $\texttt{BERT}_{\texttt{TT=10}}$ respectively in Table 3 and Table 1. We fine-tune these models with a maximum epochs of 100 and choose the best model based on the validation loss over the validation set. Similar to the case of tabular datasets, we apply `Fair-IJ` algorithm with same range of k and IHVP scaling.