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].

Figure 1: 

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 \(h_\theta(\mathbf{x})\) parameterized by a set of parameters \(\theta \in \Theta \subseteq \mathbb{R}^D\), and a loss function \(\ell : \Theta \times \mathcal{Y} \to \mathbb{R}\), we minimize the empirical risk,

\[
\hat{\theta} = \operatorname*{argmin}_{\theta \in \Theta} \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, h_\theta(\mathbf{x}_n)),
\]
to arrive at a trained predictor \( h_\theta(x) \). We will denote \( L(\theta) \equiv \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, h_\theta(x_n)) \) for notational convenience. Next, consider a weighted risk minimization problem,

\[
\hat{\theta}(w) = \arg\min_{\theta \in \Theta} \frac{1}{N} \sum_{n=1}^{N} w_n \ell(y_n, h_\theta(x_n)),
\]

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

\[
\theta(w) = \hat{\theta} + \nabla_w \theta(w) \bigg|_{w=1} (w - 1) + O((w - 1)^2),
\]

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

\[
\frac{d\theta(w)}{dw_n} \bigg|_{w=1} = -H^{-1} g_n,
\]

where \( H \equiv \nabla_\theta^2 L(\theta)|_{\theta=\hat{\theta}} \), and \( g_n \equiv \nabla_\theta \ell(y_n, h_\theta(x_n))|_{\theta=\hat{\theta}} \). Recent work \cite{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 \( \theta(w) \), we apply chain rule to arrive at,

\[
\mathcal{I}_{M,n} \equiv \frac{dM(\theta(w), w)}{dw_n} \bigg|_{w=1, \theta=\hat{\theta}} = -\nabla_\theta M(\theta(w), w) \bigg|_{w=1, \theta=\hat{\theta}}^T H^{-1} g_n,
\]

where our notation makes explicit the dependence of \( M \) on \( w \). Recent work has leveraged this machinery to approximate cross-validation \cite{16,36,15}, to interpret black-box machine learning models \cite{21}, and to assess the sensitivity of statistical analyses to training data perturbations \cite{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 \( s_n \in [k] \), i.e., \( D = \{z_n = (x_n, s_n, y_n)\}_{n=1}^{N} \), that encodes the protected group membership of the \( n \)th data instance and that we are interested in binary classification, \( Y = \{1, 0\} \). 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) \cite{3} and equality of odds (EO) \cite{17}. DP requires the classifier’s predictions to be statistically independent of the sensitive attribute, \( h_\theta(X) \perp \perp S \), where \( X \) and \( S \) are random variables representing the features and the sensitive attribute. For a binary sensitive attribute, DP implies \( P(h_\theta(X) = 1 \mid S = 1) = P(h_\theta(X) = 1 \mid S = 0) \). EO, on the other hand, requires the classifier’s predictions to be statistically independent of the sensitive attribute conditioned on the true outcome, \( h_\theta(X) \perp \perp S \mid Y \). For a binary sensitive attribute, EO implies \( P(h_\theta(X) = 1 \mid S = 1, Y = y) = P(h_\theta(X) = 1 \mid S = 0, Y = y) \) for both \( y = 0 \) and \( y = 1 \). Equality of opportunity (EQOPP) \cite{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),
\[ \Delta DP(\theta) = |P(h_\theta(X) = 1 \mid S = 1) - P(h_\theta(X) = 1 \mid S = 0)|, \]
or the absolute difference in equality of odds,
\[ \Delta EO(\theta) = \sum_{y=0}^{1} |P(h_\theta(X) = 1 \mid S = 1, Y = y) - P(h_\theta(X) = 1 \mid S = 0, Y = y)|, \]
to be close to zero while minimizing the empirical risk (Equation 1). Smooth[1] surrogates to \( \Delta DP \) and \( \Delta EO \) that are estimated from an empirical distribution are commonly used in practice [42].

\[
M_D^{\Delta DP}(\theta) = |E_{P_D(X=x|S=1)}[h_\theta(x)] - E_{P_D(X=x|S=0)}[h_\theta(x)]| \\
M_D^{\Delta EO}(\theta) = \sum_{y=0}^{1} |E_{P_D(X=x|S=0,Y=y)}[h_\theta(x)] - E_{P_D(X=x|S=1,Y=y)}[h_\theta(x)]|, \quad (6)
\]

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

### 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 [33, 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 [35], 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 an 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 attributes. 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

---

1Nearly 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 one 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 \( D_{\text{val}} = \{ x_n, s_n, y_n \}_{n=1}^{N_{\text{val}}} \) to estimate \( M^{\Delta \text{DP}}_{D_{\text{val}}} (\theta) \) and \( M^{\Delta \text{EO}}_{D_{\text{val}}} (\theta) \), we can leverage the result in Equation 5 to compute the influence of the \( n \)th training instance on \( \Delta \text{DP} \),

\[
I_{\Delta \text{DP}, n} = -\nabla_\theta M^{\Delta \text{DP}}_{D_{\text{val}}} (\theta)^T H^{-1} g_n,
\]

\[
= -\nabla_\theta \left[ \mathbb{E}_{P_D (X=x|S=1)} [h_\theta (x)] - \mathbb{E}_{P_D (X=x|S=0)} [h_\theta (x)] \right] \bigg|_{\theta=\theta} H^{-1} g_n, \tag{7}
\]

and on \( \Delta \text{EO} \),

\[
I_{\Delta \text{EO}, n} = -\nabla_\theta M^{\Delta \text{EO}}_{D_{\text{val}}} (\theta)^T H^{-1} g_n,
\]

\[
= -\nabla_\theta \frac{1}{N_{\text{val}}} \sum_{y=0}^{1} \mathbb{E}_{P_D (X=x|S=0, Y=y)} [h_\theta (x)] - \mathbb{E}_{P_D (X=x|S=1, Y=y)} [h_\theta (x)] \bigg|_{\theta=\theta} H^{-1} g_n, \tag{8}
\]

where we have used \( M^{\Delta \text{DP}}_{D_{\text{val}}} (\theta) \) and \( M^{\Delta \text{EO}}_{D_{\text{val}}} (\theta) \) to denote \( M^{\Delta \text{DP}}_{D_{\text{val}}} (\theta | w), w \big|_{w=1, \theta=\theta} \) and \( M^{\Delta \text{EO}}_{D_{\text{val}}} (\theta | w), w \big|_{w=1, \theta=\theta} \). We highlight that computing the influence of training instances on group fairness metrics requires solving a single empirical risk minimization problem to recover \( \hat{\theta} \). 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 1 is a function of \( w \). This opens up the possibility of post-hoc fairness improvement of a pre-trained \( \theta \) by searching for a weight vector \( w_{\text{fair}} \) such that \( M^b_n (\hat{\theta}_{\text{fair}}) \approx 0 \), where,

\[
\hat{\theta}_{\text{fair}} \overset{\text{def}}{=} \hat{\theta} (w_{\text{fair}}) = \hat{\theta} + \sum_{n=1}^{N} \frac{d\theta (w)}{dw_n} \bigg|_{w=1} (w_{\text{fair}} - 1),
\]

\[
= \hat{\theta} - \sum_{n=1}^{N} H^{-1} g_n (w_{\text{fair}} - 1), \tag{9}
\]

and \( w_{\text{fair}} = [w_{\text{fair}}^1, w_{\text{fair}}^2, \ldots, w_{\text{fair}}^N]^T \in \mathbb{R}^N \). We could use gradient-based methods to learn \( w_{\text{fair}} \) by optimizing a desired \( M (\theta (w), w) \) with respect to \( w \). However, computing and inverting the Hessian requires \( O (ND^2 + D^3) \) 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 \( M (\theta (w), w) \) 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 \( w \) to be binary. In Proposition 3.1 we show that we can construct \( w_{\text{fair}} \) by simply zeroing out coordinates of \( w_{\text{fair}} \) 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 \( w_{\text{fair}} \) as constructed above leads to classifiers with lower group disparities. Let \( 1 \in \mathbb{R}^N \) denote an \( N \)-dimensional vector of all ones, \( b \) denote a fairness metric, \( \nabla^b_{D_{\text{val}}} (\theta (w), w) \) denote a linearized approximation to \( M^b_{D_{\text{val}}} (\theta (w), w) \big|_{w=1} \), and \( 1[\alpha > \beta] \) denote an indicator function that takes the value one if \( \alpha > \beta \) is true and zero otherwise.
Proposition 3.1. Let \( w_{\text{fair}} \in \{0,1\}^N \) be a \( N \) dimensional binary vector such that its \( n^{th} \) coordinate is \( w^n_{\text{fair}} = 1 - I_{b,n} > 0 \), then,

\[
    w_{\text{fair}} = \text{argmin}_{w \in \{0,1\}^N} \mathcal{M}_{\text{val}}^b(\theta(w), w) - M_{\text{val}}^b(\theta(1), 1),
\]

and \( \mathcal{M}_{\text{val}}^b(\theta(w_{\text{fair}}), w_{\text{fair}}) - M_{\text{val}}^b(\theta(1), 1) \leq 0 \).

Proof. Denote \( M_{\text{val}}^b(\hat{\theta}) := M_{\text{val}}^b(\theta(1), 1) \). From a first order Taylor approximation about 1, we have,

\[
    \mathcal{M}_{\text{val}}^b(\theta(w), w) = M_{\text{val}}^b(\theta) + \sum_{n=1}^N \frac{dM_{\text{val}}^b(\hat{\theta}(w), w)}{dw_n} \bigg|_{w=1,\theta=\theta}(w_n - 1),
\]

Rearranging terms,

\[
    \mathcal{M}_{\text{val}}^b(\theta(w), w) - M_{\text{val}}^b(\hat{\theta}) = \sum_{n=1}^N \mathcal{I}_{b,n}(w_n - 1)
\]

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 \( \mathcal{I}_{b,n} > 0 \)) and the second term can be either zero (when \( w_n = 1 \)) or positive (when \( w_n = 0 \) and \( \mathcal{I}_{b,n} \leq 0 \)). \( w_{\text{fair}} \) drives the second term to zero and sets the first term to the smallest value attainable by a binary \( w \). \( \square \)

It follows that \( M_{\text{val}}^b(\theta(w_{\text{fair}}), w_{\text{fair}}) \approx \leq M_{\text{val}}^b(\theta(1), 1) \), with the inequality holding when the linearization is accurate. Finally, defining \( D_\epsilon = \{z_n \mid z_n \in \mathcal{D} \text{ and } \mathcal{I}_{b,n} > 0\} \), we arrive at the post-hoc mitigated classifier by plugging in \( w_{\text{fair}} \) from Proposition 3.1 in Equation 9

\[
    \hat{\theta}_{\text{fair}} = \theta + \sum_{m \in D_\epsilon} H^{-1}_n g_m.
\]

In Appendix A we consider an alternate \( w_{\text{fair}} \) that is guaranteed to decrease both the loss \( \ell \) 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 \( O(ND^2 + D^3) \) 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:

\[
    \hat{H}^{-1}_{n+1} = \hat{H}^{-1}_n - \frac{\hat{H}^{-1}_n \nabla \ell(y_{n+1}, h_\theta(x_{n+1})) \nabla \ell(y_{n+1}, h_\theta(x_{n+1})^T \hat{H}^{-1}_n}{\nabla \ell(y_{n+1}, h_\theta(x_{n+1})) + \hat{H}^{-1}_n \nabla \ell(y_{n+1}, h_\theta(x_{n+1}))^T},
\]

with \( \hat{H}^{-1}_0 = \lambda^{-1} I_D \), and \( \lambda \) a small positive scalar.
For computing influence functions we only need to store the product of the inverse Hessian with a vector $v$, i.e., $H^{-1}v$, which should only require $O(D)$ storage. However, if we first compute the inverse Hessian and then compute the Hessian-vector product (HVP), we would need $O(D^2)$ storage. To sidestep this issue, we develop the following coupled recurrences that only use $O(D)$ storage. We call these coupled recurrences \textit{IHVP-WoodFisher},

$$
o_{n+1} = o_n - \frac{N}{N + \nabla \ell(z_{n+1})^T o_n} o_n, \quad k_{n+1} = k_n - \frac{N}{N + \nabla \ell(z_{n+1})^T k_n} o_n, \tag{14}$$

where, we use $\ell(z_{n+1})$ as shorthand for $\ell(y_{n+1}, h_\theta(x_{n+1}))$, $o_1 = \nabla \ell(y_1, h_\theta(x_1))$, and $k_1 = v$.

\begin{proposition}
Let $o_1 = \nabla \ell(z_1)$, $k_1 = v$, and $N$ denote the number of training instances. The Hessian-vector product $H^{-1}v$ is approximated by iterating through the IHVP-WoodFisher recurrence in Equation (14) and computing $k_N$.
\end{proposition}

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

\textbf{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 $p \times p$ symmetric matrix $A$ and $p$-dimensional vectors $x$ and $y$, $x^T A y = y^T A x$. From Equation 5 the influence calculation involves computing $\nabla \ell M(\hat{\theta}, 1)^T H^{-1} g_n$ for all $n$ in the training dataset. Since is symmetric, we can equivalently compute $g_n^T H^{-1} \nabla \ell M(\hat{\theta}, 1)$. We can then run the IHVP-WoodFisher iterations to approximate $H^{-1} \nabla \ell M(\hat{\theta}, 1)$. 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 $H^{-1} \nabla \ell M(\hat{\theta}, 1)$. In contrast to other approaches to scaling up influence functions \cite{33}, our approach only requires the storage of a single $p$-dimensional vector. We call this more efficient version \textit{Fair-IJ} and is summarized in Algorithm 1.

\textbf{Most influential instances.} Our development and analysis depends on first order linear approximations of non-linear functions about 1. We expect the quality of these approximations to deteriorate further away from 1, i.e., with increasing number of instances dropped. See Theorem 1 in \cite{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, \cite{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 \textbf{D}.

\section{Experiments}

We first study our method on tabular datasets including the well-known Adult dataset \cite{13} and the recently released ACSPublicCoverage \cite{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 \cite{5}.

\subsection{Tabular Datasets}

\textbf{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 \cite{21}. 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

1: **Input:** Pre-trained model parameters $\hat{\theta}$, training set $D$, loss function $\ell$, a validation set $D_{\text{val}}$ and a smooth surrogate to the fairness metric $b \in \{\Delta \text{DP}, \Delta \text{EO}, \Delta \text{EQOPP}\}$.

2: **Calculate:** $\nabla_{\theta} M(\hat{\theta}, 1)$ using Equation 7 or Equation 8.

3: **Calculate:** $r = H^{-1} \nabla_{\theta} M(\hat{\theta}, 1)$ by setting $k_1 = \nabla_{\theta} M(\hat{\theta}, 1)$ and iterating through Equation 14 for $B$ iterations.

4: **Calculate:** the fairness influence $I_{b,n}$ of each training instance $z_n$ on $D_{\text{val}}$ by computing dot product between $g_n$ and $r$.

5: **Construct:** the set $D_-$ and denote its cardinality, $|D_-| = K$.

6: **Initialize:** $\hat{\theta}_{\text{fair}}^0 := \hat{\theta}$.

7: for $k \in [1, \ldots, K]$ do

8: **Construct:** $D_k^- = \{z_n \in D_- \mid I_{b,n} > I_{b,(K-k)}\}$, where $I_{b,(K-k)}$ denotes the $(K-k)$th order statistic of the influence scores $[I_{b,1}, \ldots, I_{b,K}]$.

9: **Calculate:** $\hat{\theta}_{\text{fair}}^k$ by replacing $D_-$ with using $D_k^-$ in Equation 12.

10: **If** $b_{D_{\text{val}}}^k(\hat{\theta}_{\text{fair}}^k) < b_{D_{\text{val}}}^k(\hat{\theta}_{\text{fair}}^{k-1})$ set $\hat{\theta}_{\text{fair}} := \hat{\theta}_{\text{fair}}^k$; else set $\hat{\theta}_{\text{fair}} := \hat{\theta}_{\text{fair}}^{k-1}$ and break out of the for loop.

11: end for

12: **Return:** fair model parameters $\hat{\theta}_{\text{fair}}$.

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.

敏感属性。在其余部分，我们仅将此子集称为 Coverage 数据集。我们随机将数据集拆分为训练/验证/测试集，具有50% / 20% / 30%的百分比，分别为每个试验。此外，对于两个数据集，我们标准化了模型的特征在我们训练方法之前，并行训练我们的基线。

我们首先训练一个1隐藏层全连接人工神经网络，具有 SeLu 激活函数和100隐藏单元。此初始模型是使用标准ERM损失与批次大小设置为256。我们使用Adam优化器与学习率设置为$10^{-4}$。我们训练ERM和基线100个epoch并选择在验证集上准确率最高的模型。我们然后使用Algorithm 1来
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, \( \hat{\theta}_{\text{fair}} \). 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 \( \hat{\theta}_{\text{fair}} \), 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) BERT\_LC: a linear classifier head, b) BERT\_NC: a non-linear classifier head and c) BERT\_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 \( \lambda \) 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
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 ($\Delta$EO), difference in demographic parity ($\Delta$DP), along with the task balanced accuracy (BA).

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


**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 $w_{\text{fair}}$

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

$$w_{\text{fair}} = 1 - I [I_{M,n} > 0] I [I_{\ell,n} > 0],$$

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

$$\hat{\theta}_{\text{fair}} = \theta + \sum_{m \in D_-} H^{-1} g_m.$$  

(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 $o_1 = \nabla_{\theta} \ell (z_1), k_1 = v$, and $N$ denote the number of training instances. The Hessian-vector product $H^{-1} v$ is approximated by iterating through the IHVP-WoodFisher recurrence in Equation 14, and computing $k_N$.

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

$$H^{-1}_{n+1} = H^{-1}_n - \frac{H^{-1}_n \nabla_{\theta} \ell (z_{n+1}) \nabla_{\theta} \ell (z_{n+1})^\top H^{-1}_n}{N + \nabla_{\theta} \ell (z_{n+1})^\top H^{-1}_n \nabla_{\theta} \ell (z_{n+1})}$$  

(17)

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

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

$$H^{-1}_{n+1} \nabla_{\theta} \ell (z_{n+1}) = H^{-1}_n \nabla_{\theta} \ell (z_{n+1}) - \frac{H^{-1}_n \nabla_{\theta} \ell (z_{n+1}) \nabla_{\theta} \ell (z_{n+1})^\top H^{-1}_n \nabla_{\theta} \ell (z_{n+1})}{N + \nabla_{\theta} \ell (z_{n+1})^\top H^{-1}_n \nabla_{\theta} \ell (z_{n+1})}$$

(18)

By substituting $H^{-1}_n \nabla_{\theta} \ell (z_{n+1})$ with $o_n$ and assuming $\nabla_{\theta} \ell (z_{n+1})$ and $\nabla_{\theta} \ell (z_{n+2})$ are close, we construct the following recurrence relation

$$o_{n+1} = o_n - \frac{o_n \nabla_{\theta} \ell (z_{n+1})^\top o_n}{N + \nabla_{\theta} \ell (z_{n+1})^\top o_n}$$

(19)

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

$$H^{-1}_{n+1} v = H^{-1}_n v - \frac{H^{-1}_n \nabla_{\theta} \ell (z_{n+1}) \nabla_{\theta} \ell (z_{n+1})^\top H^{-1}_n v}{N + \nabla_{\theta} \ell (z_{n+1})^\top H^{-1}_n \nabla_{\theta} \ell (z_{n+1})}$$

(20)

By substituting $H^{-1}_n v$ with $k_{n+1}$ and $H^{-1}_n \nabla_{\theta} \ell (z_{n+1})$ with $o_n$, we get

$$k_{n+1} = k_n - \frac{o_n \nabla_{\theta} \ell (z_{n+1})^\top k_n}{N + \nabla_{\theta} \ell (z_{n+1})^\top o_n}$$

(21)

Thus, under our assumptions, when $H_n^{-1}$ converges to $H^{-1}$, $k_n$ converges to the IHVP $H^{-1}v$. 

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

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...
We now provide additional details regarding the experiments on CivilComments datasets. Table 3 includes training the base model using ERM followed by the application of our Fair-IJ and IHVP-WoodFisher regularized using Gap Regularization (GapReg) from FairMixup (FairMixup, 2019). Hirschfeld-Gebelein-Rényi

Algorithm 2 Fair-IJ (slow)

1: **Input:** Pre-trained model parameters $\theta$, training set $D$, loss function $\ell$, a validation set $D_{\text{val}}$ and a smooth surrogate to the fairness metric $b \in \{ \Delta \text{DP}, \Delta \text{EO} \}$, $M_{D_{\text{val}}}^b$.
2: **Calculate:** $H^{-1}g_n$ for each training instance $z_n$ by setting $k_1 = g_n$ and iterating through Equation 8 for $B$ iterations.
3: **Calculate:** the fairness influence $I_{b,n}$ of each training instance $z_n$ on $D_{\text{val}}$ using Equation 7 or Equation 8.
4: **Construct:** the set $D_-$ and denote its cardinality, $|D_-| = K$.
5: **Initialize:** $\theta^0 : = \theta$
6: **for** $k \in [1, \ldots, K]$ **do**
7: **Construct:** $D_k^b = \{ z_n \in D_- \mid I_{b,n} > I_{b,(K-k)} \}$, where $I_{b,(K-k)}$ denotes the $(K-k)$th order statistic of the influence scores $[I_{b,1}, \ldots, I_{b,K}]$.
8: **Calculate:** $\hat{\theta}_{\text{fair}}^b$ by replacing $D$ with using $D_k^b$ in Equation 12.
9: **if** $b_{D_{\text{val}}}^b(\hat{\theta}_{\text{fair}}^b) < b_{D_{\text{val}}}^b(\hat{\theta}_{\text{fair}}^{b,-1})$ **set** $\hat{\theta}_{\text{fair}} : = \hat{\theta}_{\text{fair}}^b$, **else** set $\hat{\theta}_{\text{fair}} : = \hat{\theta}_{\text{fair}}^{b,-1}$ and break out of the for loop.
10: **end for**
11: **Return:** fair model parameters $\hat{\theta}_{\text{fair}}$.

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 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 scores.

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 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 scores.

D Additional Dataset, Training Details and Results

In Table 2 we provide additional information about the three datasets – Adult [6], Coverage [7], and CivilComments [8]. 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 $0 – 2000$. 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

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.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Target</th>
<th>Attribute</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adult</td>
<td>Income</td>
<td>Sex</td>
<td>21815 / 10746 / 12661</td>
</tr>
<tr>
<td>Coverage</td>
<td>Health Insurance Coverage</td>
<td>Race</td>
<td>44168 / 21755 / 32471</td>
</tr>
<tr>
<td>Civil Comments</td>
<td>Toxicity</td>
<td>Muslim</td>
<td>269038 / 45180 / 133782</td>
</tr>
</tbody>
</table>

Maximum Correlation Coefficient (HGR) dependency measure \[27\], and Fair-IJ. Reported results are for the sensitive attribute MUSLIM. They include Equality of odds (\(\Delta DO\)), demographic parity (\(\Delta DP\)), 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 BERT\(_{LC}\) (Linear Classifier) and BERT\(_{NC}\) (Non-linear Classifier) are provided for ERM, GapReg, HGR, and Fair-IJ. Both our BERT\(_{LC}\) and BERT\(_{TT}\) architectures are defined based on different classification layer(s) on top of a pre-trained BERT model. In all our experiments, we use BERT\(_{base}\) model. While BERT\(_{LC}\) uses just a single dense layer on top of the pooled vector from BERT representations, BERT\(_{NC}\) 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 \[12\], we also provide quantitative results on a variant of BERT\(_{LC}\), referred as BERT\(_{TT}\), 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
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 CiviComments on the sensitive attribute MUSLIM. We report the mean and standard deviation of difference in equality of odds ($\Delta$EO), difference in demographic parity ($\Delta$DP), along with the task balanced accuracy (BA) on 5 different seeds.

variants with 4, 8, and 10 trigger tokens which are referred as BERT$_{TT=4}$, BERT$_{TT=8}$, and BERT$_{TT=10}$ 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.