A Influence Function on Bias and Extension to DNN.

A.1 Deriving the Influence Function on Bias.

In this part, we provide detailed derivation of the influence function on bias in Eq. 5 in the main work. We first start from the influence function on parameters, we can also be referred to [1, 2].

Assuming there are n training samples $z_1, z_2..., z_n$, where $z_i = (x_i, y_i)$, and let $L(z, \theta)$ represent the loss function of sample z under the model parameters θ, then the trained $\hat{\theta}$ is given by:

$$\hat{\theta} = \text{argmin}_\theta R(\theta) = \text{argmin}_\theta \frac{1}{n} \sum_{i=1}^{n} L(z_i, \theta). \tag{1}$$

Study the impact of changing the weight of a training sample z on the model parameters θ. If we increase the weight of this sample z in the training set by ϵ, then the perturbed parameters $\hat{\theta}_{\epsilon,z}$ obtained according to ERM (Empirical Risk Minimization) will be:

$$\hat{\theta}_{\epsilon,z} = \arg\min_\theta R(\theta) + \epsilon L(z, \theta). \tag{2}$$

Define the parameter change $\Delta_\epsilon = \hat{\theta}_{\epsilon,z} - \hat{\theta}$, and note that, as $\hat{\theta}$ doesn't depend on ϵ, the quantity we seek to compute can be written in terms of it:

$$\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon} = \frac{d\Delta_\epsilon}{d\epsilon}. \tag{3}$$

Since $\hat{\theta}_{\epsilon,z}$ is a minimizer of $R(\theta)$, therefore it satisfies the first-order derivative condition, which means the first-order derivative with respect to θ is zero:

$$0 = \nabla R(\hat{\theta}_{\epsilon,z}) + \epsilon \nabla L(z, \hat{\theta}_{\epsilon,z}). \tag{4}$$

Next, since $\hat{\theta}_{\epsilon,z} \to \hat{\theta}$ as $\epsilon \to 0$, we perform a Taylor expansion of the right-hand side:

$$0 \approx \{\nabla R(\hat{\theta}) + \epsilon \nabla L(z, \hat{\theta})\} + \nabla^2 R(\hat{\theta}) + \epsilon \nabla^2 L(z, \hat{\theta}) \Delta_\epsilon,$$

where we have dropped $o(\|\Delta_\epsilon\|)$ terms. Solving for Δ_ϵ, we get:

$$\Delta_\epsilon \approx -\{\nabla^2 R(\hat{\theta}) + \epsilon \nabla^2 L(z, \hat{\theta})\}^{-1} \{\nabla R(\hat{\theta}) + \epsilon \nabla L(z, \hat{\theta})\}.$$

Since $\hat{\theta}$ minimizes R, we have $\nabla R(\hat{\theta}) = 0$. Dropping $o(\epsilon)$ terms, we have

$$\Delta_\epsilon \approx -\nabla^2 R(\hat{\theta})^{-1} \nabla L(z, \hat{\theta}) \epsilon. \tag{5}$$

Note that it is assumed that R is twice-differentiable and strongly convex in θ. we define:

$$H_{\hat{\theta}} = \nabla^2 R(\hat{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta^2 L(z_i, \hat{\theta}) \tag{6}$$

exists and is positive definite. This guarantees the existence of $H_{\hat{\theta}}^{-1}$. The final influence function can be written as:

$$I_{up,params}(z) = \left. \frac{d\hat{\theta}_{\epsilon,z_k}}{d\epsilon} \right|_{\epsilon=0} = -H_{\hat{\theta}}^{-1} \nabla_{\hat{\theta}} L(z, \hat{\theta}), \tag{7}$$

Considering $B(\hat{\theta})$ measured on any $\mathcal{A}$ with any D_{ex}, our goal is to quantify how each training point z in the training set D_{tr} contributes to $B(\hat{\theta})$. We apply the chain rule on Eq. 7:

$$I_{up,bias}(z_k, B(\hat{\theta})) = \frac{dB(\hat{\theta}_{\epsilon,z_k})}{d\hat{\theta}_{\epsilon,z_k}} \frac{d\hat{\theta}_{\epsilon,z_k}}{d\epsilon}\Big|_{\epsilon=0} = -\nabla_{\hat{\theta}} B(\hat{\theta}) H_{\hat{\theta}}^{-1} \nabla_{\hat{\theta}} L(z_k, \hat{\theta}), \tag{8}$$

Intuitively, this equation can be understood in two parts: the latter part calculates the impact of removing z on the parameters. The former part corresponds to the derivative of bias with respect to parameters, assessing how changes in parameters affect the bias. Hence, this equation quantifies the influence of removing z on the bias.

A.2 Influence at Non-Convergence

In this part, we provide the theoretical proof of the feasibility of the influence function for deep networks (non-convergent) in [1]. In the derivation of the influence function, it's assumed that $\hat{\theta}$ could be the global minimum. However, if $\hat{\theta}$ is obtained in deep networks trained with SGD in a non-convex setting, it might be a local optimum and the exact influence can hardly be computed. Here we provide the proof in [1] on how can influence function approximate the parameter change in deep networks.

Consider a training point z. When the model parameters $\tilde{\theta}$ are close to but not at a local minimum, $I_{up,params}(z)$ is approximately equal to a constant (which does not depend on z) plus the change in parameters after upweighting z and then taking a single Newton step from $\tilde{\theta}$. The high-level idea is that even though the gradient of the empirical risk at $\tilde{\theta}$ is not 0, the Newton step from $\tilde{\theta}$ can be decomposed into a component following the existing gradient (which does not depend on the choice of z) and a second component responding to the upweighted z (which $I_{up,params}(z)$ tracks).

Let $g \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta L(z_i, \tilde{\theta})$ be the gradient of the empirical risk at $\tilde{\theta}$; since $\tilde{\theta}$ is not a local minimum, $g \neq 0$. After upweighting z by ϵ, the gradient at $\tilde{\theta}$ goes from $g \mapsto g + \epsilon \nabla_\theta L(z, \tilde{\theta})$, and the empirical Hessian goes from $H_{\tilde{\theta}} \mapsto H_{\tilde{\theta}} + \epsilon \nabla_\theta^2 L(z, \tilde{\theta})$. A Newton step from $\tilde{\theta}$ therefore changes the parameters by:

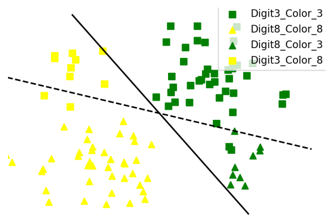
$$N_{\epsilon,z} \stackrel{\text{def}}{=} -\left[H_{\tilde{\theta}} + \epsilon \nabla_\theta^2 L(z, \tilde{\theta})\right]^{-1} \left[g + \epsilon \nabla_\theta L(z, \tilde{\theta})\right]. \tag{9}$$

Ignoring terms in ϵg, ϵ^2, and higher, we get $N_{\epsilon,z} \approx -H_{\tilde{\theta}}^{-1}\left(g + \epsilon \nabla_\theta L(z, \tilde{\theta})\right)$. Therefore, the actual change due to a Newton step $N_{\epsilon,z}$ is equal to a constant $-H_{\tilde{\theta}}^{-1} g$ (that doesn't depend on z) plus ϵ times $I_{up,params}(z) = -H_{\tilde{\theta}}^{-1} \nabla_\theta L(z, \tilde{\theta})$ (which captures the contribution of z).

B Bias Removal via Machine Unlearning

B.1 A Closer Look at the Toy Experiment

We conduct an experiment on a logistic regression task using Eq. 8. We simplify the Colored MNIST classification task to a binary classification problem of distinguishing between only digits 3 and 8, on a training set with a bias ratio of 0.95, 0.9 and 0.8, and a balanced test set. To be specific, a bias ratio of 0.95 means 95% bias-aligned samples <digit3_color3, digit8_color8> and 5% bias-conflicting samples <digit3_color8, digit8_color3> in the training set. We trained a regularized logistic regressor: $\text{argmin}_{w \in R^d} \sum_{i=1}^{n} l(w^T x_i, y_i) + \lambda \|w\|_2^2$. Fig. 1 (a) illustrates the classification results of the vanilla classifier (trained on the 0.95-biased train set) on part of test samples. We denote Digit by shape (triangle and rectangle) and Color by color (yellow and green). The solid line represents the learned classification boundary and the dotted line represents the expected classification boundary. It can be observed that the learned classifier tends to classify digits according to their color. Based on the observed bias, we employ Eq. 8 to evaluate how each training sample contributes to the bias. In Fig. 1(b), we select and visualize the most helpful (reduce bias) and harmful (increase bias) samples. We found that the most harmful samples are bias-aligned while helpful samples are bias-conflicting. With this inspiration, We further visualize the influence distribution of training samples in Fig. 2. We denote the bias-conflicting sample with "red dot" and the bias-aligned sample with "blue dot". We find that most bias-aligned samples tend to be harmful while bias-conflicting samples tend to be helpful. This pattern is consistent across different ratios of bias-conflicting samples. Additionally, the influences of helpful samples are larger than those of harmful ones. Visualizations are produced with randomly 500 samples from the training set.



(a)

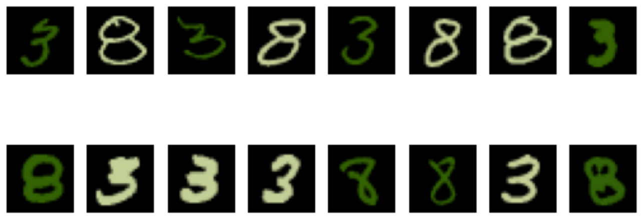
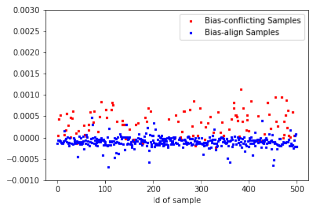
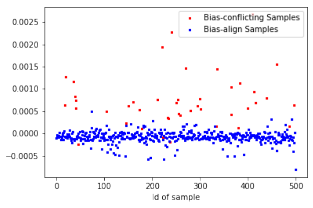

(b)

Figure 1: (a) Illustration of the learned pattern on our toy dataset. (b) Visualization of helpful samples (top row) and harmful samples (bottom row).


(a)


(b)

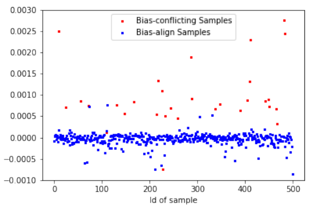
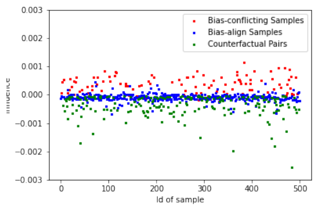

(c)

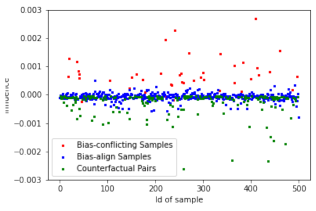
Figure 2: Influences of training samples with bias ratios of (a) 0.8, (b) 0.9, (c) 0.95.

Inspired by this observation, our unlearning strategy is further refined. Hence, we propose a straightforward solution that further mitigates the influence of a harmful sample with a bias-conflicting sample. Consequently, we update the parameters to unlearn the harmful samples by:

$$\theta_{new} = \hat{\theta} + \sum_{k=1}^{K} H_{\hat{\theta}}^{-1}(\nabla_{\hat{\theta}} L(z_k, \hat{\theta}) - \nabla_{\hat{\theta}} L(\bar{z}_k, \hat{\theta})), \tag{10}$$

where $\bar{z}_k$ denotes the bias-conflicting sample of z_k. Following the explanation in influence theory [1], our unlearn mechanism removes the effect of perturbing a training point $(\bar{a}, x, y)$ to (a, x, y). In other words, we not only remove the influence caused by harmful sample z_k, but further ensure fairness with the corresponding counterfactual sample $\bar{z}_k$.


(a)


(b)

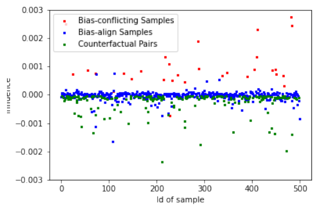

(c)

Figure 3: Influences of selected training sample (counterfactual) pairs in Eq. 10 with bias ratios of (a) 0.8, (b) 0.9, (c) 0.95.

To further illustrate the functionality of Eq. 10, we measure the influences of the selected harmful and helpful sample pairs by:

$$I_{up,bias}(z_k, B(\hat{\theta})) = -\nabla_{\hat{\theta}} B(\hat{\theta}) H_{\hat{\theta}}^{-1}(\nabla_{\hat{\theta}} L(z_k, \hat{\theta}) - \nabla_{\hat{\theta}} L(\bar{z}_k, \hat{\theta})), \tag{11}$$

with visualizations in Fig. 3. By calculating the difference between the harmful samples and helpful samples, the biased effect is significantly amplified. In this way, the unlearning becomes more effective.

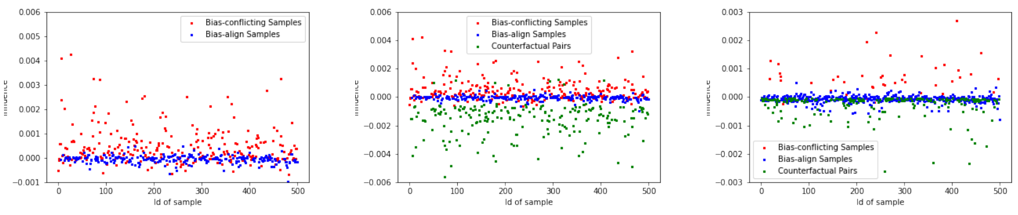
B.2 Deriving Alternative Efficient Unlearn

In the above sections, the unlearning process is based on the assumption that we could access the original training sample z_k to identify and evaluate biases and then forget them. However, in practice, the training set might be too large or even unavailable in the unlearning phase. In response, we further propose to approximate the unlearning mechanism with a small external dataset. As the influence to be removed can be obtained from the change of the protected attribute, we can construct the same modification to the protected attribute on external samples. In particular, we employ an external dataset D_{ex} as in Section 3.1 in the main work to construct counterfactual pairs for unlearning, which redefines Eq. 10 as:

$$\theta_{new} = \hat{\theta} + \sum_i H_{\hat{\theta}}^{-1} (\nabla_{\hat{\theta}} L(c_i, \hat{\theta}) - \nabla_{\hat{\theta}} L(\bar{c}_i, \hat{\theta})). \tag{12}$$

As D_{ex} can be easily obtained from an external dataset rather than the training set, e.g., the test set, the practical applicability of our method could be significantly enhanced.

We further visualize the influence of samples in the balanced external dataset in Fig. 4 (a). In the balanced dataset, the ratio of bias-aligned and bias-conflicting samples is about 50%. We can observe that the pattern of harmful bias-aligned samples and helpful bias-conflicting samples in the external dataset is similar to the training set. By comparing the influence of counterfactual pairs in the external dataset (Fig. 4 (b)) and the training set (Fig. 4 (c)), we can find the distributions are similar, which proves the feasibility of our alternative unlearning.



(a) (b) (c)

Figure 4: Influences of samples in (a) external dataset, (b) external dataset (with counterfactual sample pairs), (c) training set.

B.3 Alternative Efficient Unlearn vs. Directly Unlearn Training Data.

Tackling the problem that, in practice, the training set might be too large or even unavailable in the unlearning phase, we propose an alternative unlearning strategy in Sec. 3.3 in the main work. We approximate the change of the protected attribute by constructing the same modification to the protected attribute on external samples. Then we unlearn the same perturbation from the model with the constructed external dataset. In Sec. 4.4 in the main work, we provide the performance comparison of alternative efficient unlearn (Ours) and directly unlearn training data (Eq. 7 and Eq. 8 in the main work).

In this section, we further compare the performance of alternative unlearning on Adult and simplified Colored MNIST on logistic regression, with results reported in Tab. 1 and Tab. 2. We can find that in five experiments, alternative learning achieves comparable performance with the two directly unlearning strategies. Comparing Eq. 7 and Eq. 8, we can find that the modified Eq. 8 reaches convergence taking less iteration. The number of samples used is 200 for the two datasets.

B.4 Efficient Unlearning for Deep Networks.

In our experiment, We are inspired by Sec. 5.1 and Sec. 5.2 in [1] which keep all but the top layer in deep networks frozen and measure influence. We follow this setting so that the finetuning on deep networks can be simplified as logistic regression. In this part, we investigate the difference in finetuning different numbers of layers. The experiment is conducted on Colored MNIST with MLP with 3 hidden layers.

Attr.	Method	Bias ↓	Time(s)	# Iter.	Acc.(%) ↑
race	Vanilla	0.0134	-	-	0.8259
	Eq. 7	0.0002	1394	39	0.8249
	Eq. 8	0.0002	1398	10	0.8311
	Ours	0.0002	0.0039	46	0.8229
gender	Vanilla	0.0494	-	-	0.8259
	Eq. 7	0.0001	1386	212	0.8234
	Eq. 8	0.0001	1390	186	0.8252
	Ours	0.0006	0.0038	252	0.8232

Table 1: Alternative Efficient Unlearn on Adult.

Attr.	Method	Bias ↓	Time(s)	# Iter.	Acc.(%) ↑
0.95	Vanilla	0.4624	-	-	0.5922
	Eq. 7	0.1642	183	201	0.8548
	Eq. 8	0.1624	183	157	0.8617
	Ours	0.1496	0.0017	74	0.8594
0.9	Vanilla	0.4086	-	-	0.6517
	Eq. 7	0.1599	183	212	0.9102
	Eq. 8	0.1562	183	185	0.9211
	Ours	0.1658	0.0018	77	0.9113
0.8	Vanilla	0.3735	-	-	0.6517
	Eq. 7	0.1622	183	187	0.9241
	Eq. 8	0.1617	183	169	0.9312
	Ours	0.1611	0.0017	67	0.9244

Table 2: Alternative Efficient Unlearn on Colored MNIST.

Discussion on Different Fine-tuning Strategies. Following Sec. 4.4 in the main work, we explore the impact of unlearning different numbers of layers (i.e., the top one, two, three MLP) on the Colored MNIST with three bias ratios, with results in Tab. 3. Interestingly, the accuracy excels with two layers but decreases with three layers. Additionally, fine-tuning multiple layers takes much longer time on computation on much more parameters. It is also worth noting that our method could achieve such superior or competing performance even only by updating the last layer in deep models, which calls for more in-depth analysis in the future.

Ratio	Method	# Lay.	# Para.	Acc(%) ↑	Bias ↓	Time(s)
0.995	Vanilla	-	-	38.59	0.5863	-
	Ours^1	1	1000	62.34	0.3415	3.750
	Ours^2	2	11000	64.18	0.3378	439.34
	Ours^3	3	21000	55.32	0.3519	504.12
0.99	Vanilla	-	-	51.34	0.4931	-
	Ours^1	1	1000	71.19	0.2757	3.750
	Ours^2	2	11000	74.18	0.3134	432.86
	Ours^3	3	21000	61.45	0.2949	496.44
0.95	Vanilla	-	-	77.63	0.2589	-
	Ours^1	1	1000	86.39	0.1849	3.975
	Ours^2	2	11000	87.34	0.1902	434.25
	Ours^3	3	21000	86.47	0.1914	501.24

Table 3: Ablation on # MLP Layers.

B.5 Effectiveness of Pre-calculating Hessian.

In Sec. 3.4 in the main work, we propose to pre-calculate the inverse Hessian before performing unlearning. In this way, we approximate the Hessian as it should change with model parameters, however, we prevent the large computation cost of updating and inverting the Hessian at every iteration. In this part, we empirically illustrate the effectiveness of our approximation. Experiments are conducted on Colored MNIST and Adult datasets with logistic regression tasks, with results provided in Tab. 4 and Tab. 5. "wo/" denotes unlearning without pre-calculation. It can be observed that unlearning with or without can achieve comparative performance on bias and accuracy. However, our method can save about 40% run time on Adult and 97% run time on Colored MNIST. The reason is that the number of parameters for Colored MNIST is much larger than Adult, so that the calculation of inverse Hessian makes up a larger proportion of the total run time.

Attr.	Method	Bias ↓	Time(s)	# Iter.	Acc.(%) ↑
race	Vanilla	0.0134	-	-	0.8259
	wo/	0.0002	0.0064	42	0.8229
	Ours	0.0002	0.0039	46	0.8229
gender	Vanilla	0.0494	-	-	0.8259
	wo/	0.0006	0.0066	149	0.8243
	Ours	0.0006	0.0038	252	0.8232

Table 4: Efficient Hessian Computation on Adult.

Ratio	Method	Bias ↓	Time(s)	# Iter.	Acc.(%) ↑
0.95	Vanilla	0.4624	-	-	0.5922
	wo/	0.1490	0.0556	59	0.8674
	Ours	0.1496	0.0017	74	0.8594
0.9	Vanilla	0.6517	-	-	0.4086
	wo/	0.1698	0.0498	46	0.9093
	Ours	0.1658	0.0018	77	0.9113
0.8	Vanilla	0.2857	-	-	0.6915
	wo/	0.1689	0.0517	34	0.9264
	Ours	0.1611	0.0017	67	0.9244

Table 5: Efficient Hessian Computation on Colored MNIST.

C Experiment Details

C.1 Dataset

Colored MNIST. Colored MNIST is constructed based on the MNIST dataset [3] designed for digit classification tasks. To build a biased correlation, ten distinct RGB values are applied on grayscale digit images [4, 5, 6]. Digit and color distribution are paired to build biased correlations in the training set. Bias-aligned samples are defined as fixed combinations of digit and color like Digit 1, Color 1 while bias-conflict samples are defined as other combinations like Digit 1, random Color in 2-10. In our Experiment, we use 3 different training sets by setting different bias ratios 0.995, 0.99, 0.95 for biased-aligned training samples where the ratio represents the partition of bias-aligned samples in the training set. The higher the ratio, the higher the degree of bias. The split of the training set, test set, and external set is 60000, 10000, and 10000.

CelebA. CelebA dataset [7] is a face recognition with 40 types of attributes like gender, age (young or not), and lots of facial characteristics (such as hair color, smile, beard). The dataset contains a total of 202,599 images which, following the official train validation split, consists of 162,770 images for training and 9,867 images for testing. We choose Gender as the protected attribute, Hair-color (blonde hair or not) and Attractive as the target attribute following [8, 9]. The number of selected samples for the two target attributes is 200 and 182, which are split from the test set.

Adult Income Dataset. The Adult dataset is a publicly available dataset in the UCl repository [10] based on 1994 U.S. census data. The goal of this dataset is to successfully predict whether an individual earns more or less than \$50,000 per year based on features such as occupation, marital status, and education. We follow the processing procedures in [11]. In our experiment, we choose gender and race as protected attributes following [12, 13]. We split 200 samples from the test set as the external dataset.

C.2 Baselines

For the sanity check experiment on a toy Colored MNIST dataset, we use a vanilla logistic regression model as the baseline. For experiments with deep networks, we compare our method with one pre-processing baseline Reweigh [14], 6 in-processing debiasing baselines (LDR [15], LfF [16], Rebias [17], DRO [8], SenSEI [18], and SenSR [19]) and 4 post-processing baselines (EqOdd [20], CEqOdd [20], Reject [21] and PP-IF [22]). [14] utilizes the influence function to reweight the training sample, in order to re-train a fair model targeting group fairness metrics (equal opportunity and demographic parity). Among in-processing baselines, LDR, LfF, Rebias, and DRO are designed explicitly to target higher accuracy (on unbiased test set or worst-group test set) and implicitly target fairness, while SenSEI and SenSR are designed to target individual fairness. EqOdd, CEqOdd and Reject are designed to target different group fairness metrics (equal odd and demographic parity), while [22] proposes a post-processing algorithm for individual fairness.

D Discussion

D.1 Dataset Generation

In our experiments, we utilize approximated counterfactual samples for CelebA due to the unavailability of strict counterfactual data. Based on attribute annotations, we select images with the same target attributes but opposite sensitive attributes, while maintaining other attributes as much as possible. Our method achieves the best results on the worst-case group, indicating that the approximated counterfactual samples can also effectively enhance fairness in predictions. Similar to our approach, [23] proposes to select pairs of counterfactual images based on attribute annotations on the CUB dataset to produce counterfactual visual explanations. Their experiments also show that neural networks can discern major differences (such as gender in our work) between images without strict control (such as background).

For real-world visual datasets (like facial dataset or ImageNet), the unavailability of strict counterfactual data is a common challenge. Existing methods propose to train a generative model to create counterfactual images with altered sensitive attributes [24, 25, 26], which seems to be a viable approach for obtaining counterfactual datasets for more diverse vision applications. Building upon these methods, we will extend our approach to more scenarios.

D.2 Influence Estimation

In our unlearning experiment, we freeze the parameters of all other layers except the top layer. Previous work investigates the estimation accuracy of the influence function on both multi-layer and single-layer setups [27]. It performs a case study on the MNIST. For each test point, they select 100 training samples and compute the ground-truth influence by model re-training. Results show that estimations are more accurate for shallow networks.

Our results in Tab. 7 in the main manuscript also validate this point. When applying FMD to a three-layer neural net, the performance on either accuracy or bias becomes worse. This could potentially be attributed to the inaccurate estimation of influence function on multi-layer neural nets. In our experiments, we adhere to the set-up in [1], where the influence function is only applied to the last layer of deep models, which proves to be effective.

As verified in [1, 27, 28], influence estimation matches closely to leave-one-out retraining for logistic regression model. As discussed in [24], measuring influence score for the last layer can be regarded as calculating influence from a logistic regression model on the bottleneck features (Sec. 5.1 in the main manuscript). The same setup is followed by many influence function-based works [29, 30] and proves to be effective.

D.3 Computational Complexity

As for bias-effect evaluation, with n training points and $\theta \in R^d$, directly computing Eq. 5 (in the main manuscript) requires $O(nd^2 + nd^3)$ operations. In our experiment, we only activate the last layer so that d is small. However, when the number of training samples is very large, performing Eq. 5 is expensive. As for the debiasing phase, it requires $O(nd^2 + kd^2)$ operations, where k is the number of samples to unlearn. Note that if hessian is calculated in the bias-effect evaluation phase, it can be directly used in the debiasing phase. Hence, the overall computational complexity using Eq. 7 and Eq. 8 is $O(nd^2 + kd^2 + nd^3)$.

However, in our proposed alternative debiasing method, we only utilize an external counterfactual dataset with a small number of k. Hence, we can omit the $O(nd^3)$ operations to compute influences and rank the training samples. Hence, the overall computational complexity using Eq. 9 (Ours) is $O(nd^2 + kd^2)$. Experimental comparison results can be referred to Tab. 5 (in the main manuscript). Debiasing with Eq. 8 takes about 500x more time than Eq. 9 (in the main manuscript).

E Preliminaries

E.1 Influence Function

The origins of influence-based diagnostics can be traced back to important research papers such as [2, 31, 32]. More recently, Koh and Liang [1] introduced the concept of influence functions to large-scale deep learning, which numerous publications have since followed up. In their work, [1] advocated for the use of an approximation, Eq. 2, to estimate the change in loss when a small adjustment is made to the weights of the dataset. In practical applications involving deep models, the Hessian matrix (H) cannot be stored in memory or inverted using standard linear algebra techniques. However, by considering a fixed vector (v), the Hessian vector product (HVP), Hv, can be computed in O(bp) time and memory [33], where b represents the batch size and determines the number of training examples used to approximate H (for a given loss function L). The iterative procedure LISSA [34], employed by [1], relies on repeated calls to the HVP to estimate the inverse HVP.

E.2 Counterfactual Fairness

Counterfactual fairness, a relatively new concept, has emerged as a means to measure fairness at an individual level [35]. The fundamental idea behind this approach is to determine the fairness of a decision for an individual by comparing it with the decision that would have been made in an alternate scenario where the individual's sensitive attributes possessed different values. This concept builds upon earlier work [36], which introduced a causal framework for learning from biased data by examining the relationship between sensitive features and the data. Recent advancements in deep learning have further contributed to this field, with novel approaches [37, 38, 39, 25] proposing methods to enhance the accuracy of decision-making models by improving the approximation of causal inference, particularly when dealing with unobserved confounding variables.

E.3 Demographic Parity and Equal Opportunity

Demographic Parity [40]: A predictor Y satisfies demographic parity if $P(Y|A=0) = P(Y|A=1)$, where A is the sensitive attribute. The likelihood of a positive outcome should be the same regardless of whether the person is in the protected (e.g., female) group.

Equal Opportunity [20]: "A binary predictor Y satisfies equal opportunity with respect to A and Y if $P(Y=1|A=0,Y=1) = P(Y=1|A=1,Y=1)$ " . This means that the probability of a person in a positive class being assigned to a positive outcome should be equal for both protected and unprotected (female and male) group members.

References

[1] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR, 2017.

[2] R Dennis Cook and Sanford Weisberg. *Residuals and influence in regression*. New York: Chapman and Hall, 1982.

[3] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *IEEE*, 86(11):2278–2324, 1998.

[4] Byungju Kim, Hyunwoo Kim, Kyungsu Kim, Sungjin Kim, and Junmo Kim. Learning not to learn: Training deep neural networks with biased data. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[5] Yi Li and Nuno Vasconcelos. Repair: Removing representation bias by dataset resampling. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[6] 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*, 2020.

[7] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *IEEE International Conference on Computer Vision*, 2015.

[8] Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. In *International Conference on Learning Representations*, 2020.

[9] Enzo Tartaglione, Carlo Alberto Barbano, and Marco Grangetto. End: Entangling and disentangling deep representations for bias correction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13508–13517, June 2021.

[10] Andrew Frank, Arthur Asuncion, et al. Uci machine learning repository, 2010. *URL http://archive. ics. uci. edu/ml*, 15:22, 2011.

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

[12] Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. Learning fair representations. In *International conference on machine learning*, pages 325–333. PMLR, 2013.

[13] Toshihiro Kamishima, Shotaro Akaho, and Jun Sakuma. Fairness-aware learning through regularization approach. In *2011 IEEE 11th International Conference on Data Mining Workshops*, pages 643–650. IEEE, 2011.

[14] Peizhao Li and Hongfu Liu. Achieving fairness at no utility cost via data reweighing with influence. In *International Conference on Machine Learning*, pages 12917–12930. PMLR, 2022.

[15] Jungsoo Lee, Eungyeup Kim, Juyoung Lee, Jihyeon Lee, and Jaegul Choo. Learning debiased representation via disentangled feature augmentation. *Advances in Neural Information Processing Systems*, 34:25123–25133, 2021.

[16] Junhyun Nam, Hyuntak Cha, Sungsoo Ahn, Jaeho Lee, and Jinwoo Shin. Learning from failure: Training debiased classifier from biased classifier. In *Advances in Neural Information Processing Systems*, 2020.

[17] 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 (ICML)*, 2020.

[18] Mikhail Yurochkin and Yuekai Sun. Sensei: Sensitive set invariance for enforcing individual fairness. *arXiv preprint arXiv:2006.14168*, 2020.

[19] Mikhail Yurochkin, Amanda Bower, and Yuekai Sun. Training individually fair ml models with sensitive subspace robustness. *arXiv preprint arXiv:1907.00020*, 2019.

[20] Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. *Advances in neural information processing systems*, 29, 2016.

[21] Faisal Kamiran, Asim Karim, and Xiangliang Zhang. Decision theory for discrimination-aware classification. In *2012 IEEE 12th International Conference on Data Mining*, pages 924–929, 2012.

[22] Felix Petersen, Debarghya Mukherjee, Yuekai Sun, and Mikhail Yurochkin. Post-processing for individual fairness. *Advances in Neural Information Processing Systems*, 34:25944–25955, 2021.

[23] Yash Goyal, Ziyan Wu, Jan Ernst, Dhruv Batra, Devi Parikh, and Stefan Lee. Counterfactual visual explanations. In *International Conference on Machine Learning*, pages 2376–2384. PMLR, 2019.

[24] Saloni Dash, Vineeth N Balasubramanian, and Amit Sharma. Evaluating and mitigating bias in image classifiers: A causal perspective using counterfactuals. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 915–924, 2022.

[25] Hyemi Kim, Seungjae Shin, JoonHo Jang, Kyungwoo Song, Weonyoung Joo, Wanmo Kang, and Il-Chul Moon. Counterfactual fairness with disentangled causal effect variational autoencoder. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8128–8136, 2021.

[26] Jiaee Cheong, Sinan Kalkan, and Hatice Gunes. Counterfactual fairness for facial expression recognition. In *European Conference on Computer Vision*, pages 245–261. Springer, 2022.

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

[28] Juhan Bae, Nathan Ng, Alston Lo, Marzyeh Ghassemi, and Roger B Grosse. If influence functions are the answer, then what is the question? *Advances in Neural Information Processing Systems*, 35:17953–17967, 2022.

[29] Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33:19920–19930, 2020.

[30] Chih-Kuan Yeh, Joon Kim, Ian En-Hsu Yen, and Pradeep K Ravikumar. Representer point selection for explaining deep neural networks. *Advances in neural information processing systems*, 31, 2018.

[31] R Dennis Cook and Sanford Weisberg. Characterizations of an empirical influence function for detecting influential cases in regression. *Technometrics*, 22(4):495–508, 1980.

[32] R Dennis Cook and Sanford Weisberg. *Residuals and influence in regression*. New York: Chapman and Hall, 1982.

[33] Barak A Pearlmutter. Fast exact multiplication by the hessian. *Neural computation*, 6(1):147–160, 1994.

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

[35] Matt J Kusner, Joshua Loftus, Chris Russell, and Ricardo Silva. Counterfactual fairness. *Advances in neural information processing systems*, 30, 2017.

[36] Judea Pearl et al. Models, reasoning and inference. *Cambridge, UK: CambridgeUniversityPress*, 19(2), 2000.

[37] Yongkai Wu, Lu Zhang, and Xintao Wu. Counterfactual fairness: Unidentification, bound and algorithm. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 2019.

[38] Sahaj Garg, Vincent Perot, Nicole Limtiaco, Ankur Taly, Ed H Chi, and Alex Beutel. Counterfactual fairness in text classification through robustness. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 219–226, 2019.

[39] Stephen R Pfohl, Tony Duan, Daisy Yi Ding, and Nigam H Shah. Counterfactual reasoning for fair clinical risk prediction. In *Machine Learning for Healthcare Conference*, pages 325–358. PMLR, 2019.

[40] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 214–226, 2012.