RULER: Discriminative and Iterative Adversarial Training for Deep Neural Network Fairness

Guanhong Tao*
taog@purdue.edu
Purdue University
USA

Weisong Sun*
weisongsun@smail.nju.edu.cn
State Key Laboratory for Novel Software Technology
Nanjing University
China

Tingxu Han*
txhan@smail.nju.edu.cn
School of Information Management
Nanjing University
China

Chunrong Fang†
fangchunrong@nju.edu.cn
State Key Laboratory for Novel Software Technology
Nanjing University
China

Xiangyu Zhang
xyzhang@cs.purdue.edu
Purdue University
USA

ABSTRACT

Deep Neural Networks (DNNs) are becoming an integral part of many real-world applications, such as autonomous driving and financial management. While these models enable autonomy, there are however concerns regarding their ethics in decision making. For instance, fairness is an aspect that requires particular attention. A number of fairness testing techniques have been proposed to address this issue, e.g., by generating test cases called individual discriminatory instances for repairing DNNs. Although they have demonstrated great potential, they tend to generate many test cases that are not directly effective in improving fairness and incur substantial computation overhead. We propose a new model repair technique, Ruler, by discriminating sensitive and non-sensitive attributes during test case generation for model repair. The generated cases are then used in training to improve DNN fairness. Ruler balances the trade-off between accuracy and fairness by decomposing the training procedure into two phases and introducing a novel iterative adversarial training method for fairness. Compared to the state-of-the-art techniques on four datasets, Ruler has 7-28 times more effective repair test cases generated, is 10-15 times faster in test generation, and has 26-43% more fairness improvement on average.

CCS CONCEPTS

• **Software and its engineering** → **Software testing and debugging**; • **Computing methodologies** → *Neural networks.*

*Equal contribution
†Corresponding author

KEYWORDS

Deep Neural Network, Fairness, Adversarial Training

ACM Reference Format:
Guanhong Tao, Weisong Sun, Tingxu Han, Chunrong Fang, and Xiangyu Zhang. 2022. RULER: Discriminative and Iterative Adversarial Training for Deep Neural Network Fairness. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '22), November 14–18, 2022, Singapore, Singapore.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3540250.3549169

1 INTRODUCTION

Deep Neural Networks (DNNs) are being widely employed in real-world applications, such as face recognition [10], speech recognition [11], autonomous driving [12]. A criminal identification system may determine whether a suspect is an actual criminal based on the background information of the suspect, such as prior record, race, gender, etc. A financial institution may decide whether to authorize a loan to a customer based on one's personal information. Fairness in such decisions is hence particularly important. For instance, a criminal identification system with biases towards a certain race would have detrimental societal ramifications.

The fairness problems in DNNs can be the discrimination against certain protected or sensitive attributes, such as race, gender, etc [6]. That is, a DNN model's prediction is inappropriately tied with specific values of protected/sensitive attributes (e.g., gender) [2, 45]. The bias/unfairness of DNNs can also lie in the prediction outcome differences towards different labels, such as divergent predication errors for different persons in face identification. There exist a number of fairness types in the literature, such as *individual discrimination* [14], *group discrimination* [15], etc. Different fairness problems require specific designs to achieve the mitigation goal. In this paper, we focus on the individual discrimination problem following existing works [43, 45]. Specifically, given an input x, there exists another sample x' that has different values on sensitive attributes but the same values on non-sensitive ones. Sample x' is an *individual discriminatory instance* (IDI) if the model produces different predictions for x and x'. Addressing the individual discrimination problem is hence to reduce the number of such cases for a given model.

In order to build a fair DNN model, existing techniques aim to generate IDIs [1, 2, 4, 16] and some use generated IDIs to repair the model [14, 37, 42, 43]. THEMIS [16] randomly selects values from the valid ranges of all attributes to generate IDIs. AEQUITAS [37] first exhaustively enumerates all the values for sensitive attributes to generate random samples. It then perturbs other attributes of each random sample. To improve the diversity of IDIs, SG [2] leverages dynamic symbolic execution to generate IDIs. The aforementioned methods, namely, THEMIS, AEQUITAS, and SG, are mainly designed for traditional machine learning models, such as logistic regression [26], support vector machine [8], and decision tree [9], etc. They may be inapplicable to DNNs or too expensive as DNNs are more complex. Recent work leverages gradient information from DNN models to better assist fairness improvement (e.g., ADF [43], EIDIG [42], and NeuronFair [45]). They mainly follow a similar instance generation procedure by adversarially perturbing input attributes and leveraging majority voting [25] to assign labels for IDIs during retraining. The magnitude of adversarial perturbation on those attributes is usually not bounded, which can lead to illegitimate samples. Existing techniques strictly follow the definition of individual discrimination (details in Section 2.2) during instance generation and only perturb one sensitive attribute at a time. This incurs substantial computation overhead and may generate samples that are not IDIs. In addition, existing techniques are not iterative in model repair due to their high cost. They hence may not achieve the optimal fairness improvement.

Adversarial training is one of the most effective techniques in constructing robust models [28, 36]. It iteratively generates adversarial examples with respect to the current state of the model at each training step. It aims to flip the prediction of an input by adding a small perturbation on the input. A straightforward idea is to directly leverage adversarial training for improving fairness. We observe that adversarial training can address the problem but at the cost of non-trivial accuracy degradation. It is known that adversarial training affects normal functionalities [28].

We propose a novel fair model training method, called Ruler, which is specially designed for improving fairness. It decomposes the training procedure into two phases: *warm-up* and *fairness adversarial training*. The first phase follows the normal training to ensure the functionalities (i.e., model accuracy). A validation set is utilized to determine whether a model has satisfactory performance on normal samples. In the fairness adversarial training phase, Ruler iteratively generates IDIs for repair. Different from existing fairness improving techniques that strictly follow the definition of individual discrimination by requiring a coupling sample that only differs on sensitive attributes from an input when perturbing sensitive and non-sensitive attributes, Ruler relaxes the conditions by allowing small perturbations on non-sensitive attributes without coupling samples. Although perturbing non-sensitive attributes has been explored before, the relaxation in Ruler and the small bound on non-sensitive attributes substantially enlarges the chance of generated samples being real IDIs. Specifically, we impose different bounds on sensitive and non-sensitive attributes, where sensitive attributes shall be within their valid value ranges, and non-sensitive ones shall be in the neighborhood of the original input (bounded by a small L_∞). Ruler also introduces a queue to ensure that all the original training samples are utilized in each adversarial training epoch to prevent model accuracy degradation. By balancing the number of normal samples and IDIs during repair, Ruler achieves better accuracy and fairness compared to the state-of-the-art. In this paper, we mainly focus on tabular data with explicitly extracted features. Ruler can be possibly extended to other domains, which is discussed in Section 6.

Our contributions are summarized as follows.

- We relax the conditions in individual discrimination definition and allow small perturbations on non-sensitive attributes without coupling samples. It leads to more effective and efficient IDI generation.
- We formulate the fairness improvement task as a min-max optimization problem by adversarially and iteratively training on generated IDIs. It also features a warm-up phase and a sample fusion technique to retain normal functionalities of the repaired model.
- We develop a prototype Ruler and evaluate it on four widely used public datasets. The experimental results demonstrate that Ruler has 97.86% relative fairness improvement with only 0.55% accuracy degradation on average. We compare with three state-of-the-art fairness improvement techniques and show that Ruler is 10-15 times faster in generating IDIs and has 26-43% more fairness improvement. The implementation is publicly available [35].

2 BACKGROUND

In this section, we briefly review relevant background, including DNN, individual discrimination, and adversarial attack. We then define our problem.

2.1 Deep Neural Network

A DNN can be represented as a function $f : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ is the input space and $\mathcal{Y}$ the output space. A DNN model usually consists of a sequence of n layers that are connected as follows.

$$f_\theta(x) = f_{n-1} \circ f_{n-2} \cdots \circ f_0(x), \tag{1}$$

where f_0 is the first layer and f_{n-1} the last. Variable θ denotes all the weight parameters in the n layers. To train a model that can correctly predict output $y \in \mathcal{Y}$ given an input $x \in \mathcal{X}$, a loss function $\mathcal{L}$ is utilized for searching the optimal parameters θ such that the empirical risk is minimized as follows.

$$\arg\min_\theta \underset{(x,y)\sim\{\mathcal{X},\mathcal{Y}\}}{\mathbb{E}} \Big[\mathcal{L}\big(f_\theta(x), y\big)\Big]. \tag{2}$$

The empirical risk is the expectation of the loss on all the samples in the given training set as shown above. The goal is to obtain a set of parameters θ that has the smallest empirical risk, which is supposed to correctly predict unseen inputs, i.e., those from the test set. The prediction accuracy on the test set denotes the functionality of f. The higher the test accuracy is, the better a DNN model is.

2.2 Individual Discrimination

Individual discrimination describes the scenario where a decision-making outcome is inappropriately tied with a specific value of some sensitive/protected attribute, such as gender and race. In this paper, we follow the formal problem definition introduced in the literature [42, 43, 45].

Let $\mathcal{X} \in \mathbb{R}^{n \times d}$ be the input space, where n is the number of input samples and d the number of attributes. We use $A = \{a_i | i \in \{1, \cdots, d\}\}$ to denote the attributes. Suppose each attribute $a_i \in \mathbb{I}_i$ (valid value domain), then $\mathbb{I} = \mathbb{I}_1 \times \mathbb{I}_2 \times \cdots \times \mathbb{I}_d$ denotes all possible combinations of attribute values. Let $P \subset A$ denote a set of sensitive/protected attributes, and $A \backslash P$ the set of non-sensitive attributes. An individual discriminatory instance for a DNN model f trained on $\mathcal{X}$ is defined as follows.

Definition 1 (Individual Discriminatory Instance). *Let $x = \{x_1, x_2, \cdots, x_d\}$ be an arbitrary instance in $\mathbb{I}$, where x_i is the value of the i-th attribute, i.e., a_i. Instance x is considered an individual discriminatory instance for model f if there exists an instance $x' \in \mathbb{I}$ that satisfies the following conditions:*

- $\exists p \in P, x_p \neq x'_p$
- $\forall q \in A \backslash P, x_q = x'_q$
- $f(x) \neq f(x')$

Pair (x, x') denotes an individual discriminatory instance pair. The above conditions are strict and defined on the whole value space $\mathbb{I}$. In real world scenarios, the input space $\mathcal{X}$ is a subset of all possible instances in the value range $\mathbb{I}$. There may not exist an x' that satisfies the above conditions. To explore more possible individual discriminatory instances, we relax the conditions during instance generation, which is discussed in Section 4.2.

2.3 Adversarial Attack

Adversarial attacks craft human-imperceptible small perturbations that can induce misclassification on DNNs when added to normal inputs [33]. It has been extensively studied in the literature [3, 13, 20, 21, 38, 40, 44]. A pioneer work by Goodfellow et al. [33] introduces Fast Gradient Sign Method (FGSM) that leverages the gradient sign to perturb an input. Specifically, it computes the gradient of a loss function with respect to the input and adds the sign value of the gradient (with some proportion) onto the input. Given an input x and a subject model loss function $\mathcal{L}$ (e.g., cross entropy loss), an adversarial example is crafted as follows.

$$x^{adv} = x + \epsilon \cdot \text{sign}\Big(\nabla_x \mathcal{L}\big(f_\theta(x), y\big)\Big), \tag{3}$$

where ϵ is a hyper-parameter determining the magnitude of added perturbation; sign$(\cdot)$ takes the sign of input values (e.g., -1 for value -2.5); $\mathcal{L}$ is the loss function of subject model f_θ with θ the weight parameters and y the ground truth label; ∇_x computes the gradient of $\mathcal{L}$ with respect to the input x. FGSM uses one step of gradient information according to the loss function to construct adversarial examples. The perturbation magnitude is constrained by the hyper-parameter ϵ such that $||x^{adv} - x||_\infty \leq \epsilon$.

Advanced adversarial attacks extend FGSM for leveraging the gradient information via multiple steps [7, 24, 27]. Projected Gradient Descent (PGD) [28] is one of such variants. It also projects the perturbed input onto the valid input space (i.e., the legitimate value range) at each step during adversarial generation. Formally,

$$x^{t+1} = \prod_{x+\mathcal{S}} \left(x^t + \alpha \cdot \text{sign}\Big(\nabla_x \mathcal{L}\big(f_\theta(x), y\big)\Big)\right), \tag{4}$$

where x^{t+1} is the modified input at step $t+1$; $\mathcal{S}$ is a set of allowed perturbations within the valid input space; $\prod$ denotes the projection to valid space $x + \mathcal{S}$; Variable α is a hyper-parameter controlling the perturbation magnitude at each step.

2.4 Problem Statement

Existing DNNs have undesirable discrimination against certain sensitive attributes. That is, the prediction of a model can be flipped when only some sensitive attribute of an input is altered. This leads to unfair decision-makings when such models are used in critical systems, such as criminal identification system. In this paper, we aim to repair DNNs by mitigating the discrimination rooted in models. A repaired DNN model shall produce the same prediction if two inputs only differ on sensitive attributes. Given an input $x \in \mathcal{X}$ and a model f, the fairness goal is as follows.

$$\forall x',\ \forall p \in P, x_p \neq x'_p,\ \forall q \in A - p, x_q = x'_q,\ f(x) = f(x'), \tag{5}$$

where $p \in P$ is a sensitive attribute and $A - p$ the attributes other than p. As it is infeasible of having the whole input distribution, a sampled set $\mathcal{X}$ is commonly used in practice to evaluate the performance of f. For every sample $x \in \mathcal{X}$, a validation tool evaluates the fairness of model f according to Equation 5. The details are elaborated in Section 5.2. The more samples satisfy the goal, the fairer a model is on sensitive attributes.

3 MOTIVATION

Effectively generating individual discriminatory instances is the first step towards mitigating unfairness in DNNs. Existing techniques [42, 43, 45] mainly follow a similar instance generation procedure by (1) adversarially perturbing input features and (2) leveraging majority voting [25] to assign labels for a set of randomly selected discriminatory instances. In the first phase of adversarial perturbation, existing techniques do not constrain the magnitude of the perturbation. That is, any input feature dimensions can be altered to arbitrary values (e.g., changing age 50 to 10) as long as they can induce the intended output, i.e., flipping the predicted label (e.g., from authorizing to NOT authorizing a loan). As the prediction changes when the sensitive attribute is altered, this sample x' (from the original input x) might be considered an individual discriminatory instance (IDI). Hence, to improve the model fairness, this sample x' will be added to the original training dataset by retaining its original label y (e.g., authorizing a loan). However, there is no guarantee for those generated instances to be legitimate as they may not follow the normal data distribution (e.g., authorizing a loan to an age 10). To determine whether the generated samples are indeed IDIs, existing techniques adopt a majority voting process. Specifically, they first train a set of models on the original training dataset without the sensitive attributes and then feed the generated samples to obtain predictions. The final label z is determined by the majority voting on these predictions. As the majority voting models are not trained on sensitive attributes, they may produce a legitimate label (of not authorizing a loan to an age 10). Ideally, all the generated samples shall be IDIs, whose labels shall be the same as the original labels (before perturbations) according to Equation 5, i.e. $z = y$. However, we find that more than 40% of the generated samples by the state-of-the-art techniques ADF [43]

Table 1: Unfairness of models trained with different strategies. Sen. Att. denotes sensitive attributes. OL denotes using the original labels of the generated samples.

Sen. Att.	Original	ADF	ADF OL	EIDIG	EIDIG OL
Age	11.7%	4.45%	27.7%	2.28%	24.8%
Race	9.9%	2.61%	24.5%	1.00%	18.9%
Gender	3.6%	1.24%	12.1%	0.89%	9.6%
Accuracy	84.32%	83.66%	82.64%	84.12%	82.85%

and EIDIG [42] have labels (by voting) different from their original labels ($z \neq y$), meaning those samples are not real IDIs but some arbitrary samples. As a result, using these samples in training can hardly improve fairness. In addition, existing techniques strictly follow Definition 1 when generating IDIs. That is, the adversarial perturbation ought to have the same gradient direction (the value sign) on non-sensitive attributes for the coupling pair x and x' such that they only differ on sensitive attributes. They also only generate instances for one single sensitive attribute at a time. This induces high computation costs during generation. They often take minutes to produce one thousand instances (10x times slower than Ruler) as shown in Section 5.3.

We conduct a study to validate the discrimination of generated samples by existing techniques, that is, using the original labels instead of the majority voted ones as the labels to train. We then train the subject model with these samples following the same procedure in [42, 43]. We use the Adult dataset [22] as an example. Table 1 shows the unfairness of different models. Rows 2-4 show the model unfairness on three sensitive attributes (Sen. Att.), i.e., Age, Race, and Gender. The smaller the value, the fairer the model on sensitive attributes. The last row denotes model accuracy on the test set (the higher, the better). Column 2 shows the results of the original DNN model without any discrimination mitigation techniques. Columns 3 and 5 show the results of two state-of-the-art techniques ADF [43] and EIDIG [42]. Columns 4 and 6 present the results of removing the majority voting component in ADF and EIDIG and using the original labels as the ground truth (ADF OL and EIDIG OL). Observe that both ADF and EIDIG have reasonable performances in improving fairness. However, when considering all the generated samples as IDIs by assigning them their original labels, the fairness on all three sensitive attributes becomes substantially worse. The test accuracies are also lower. This validates our hypothesis that a large number of generated samples by existing techniques are in fact not IDIs. It undermines the effectiveness and efficiency of existing techniques as we will demonstrate in Section 5.3.

The ultimate goal of generating discriminatory instances is to construct DNN models with improved fairness. Existing techniques usually randomly select 5-10% of their generated instances and add them to the original training dataset for retraining [42, 43, 45]. Such a procedure only focuses on the discriminatory cases regarding the original model. Just like adversarial examples can still be generated for the model retained on a set of previously generated adversarial examples, discriminatory instances still exist in the models fixed by existing techniques. Iteratively applying existing techniques for retraining the model is very expensive as many generated cases are non-discriminatory, and the cost is high as discussed earlier. Adversarial training generates adversarial examples with respect to the current state of the model at each training step. It aims to flip the prediction of an input by adding a small perturbation to the input. The perturbation on every dimension of the input is bounded by the same L_p norm. A straightforward idea is to directly leverage adversarial training for improving fairness. We employ a well-known adversarial training approach PGD [28] and show the results in Figure 1. The x-axis denotes the training epoch, and the y-axes denote the validation accuracy on the left and unfairness on the right. The red triangle line (Acc_PGD) shows the accuracy and the red circle line (Unfair_PGD) the unfairness for PGD-trained model. Observe that adversarial training can reduce the discriminatory instances to almost 0%. The accuracy, however, is only 76.56%, much lower than that of a naturally trained model (84.32%). It is known that adversarial training leads to non-trivial accuracy degradation [28], which is undesirable for normal functionalities.

Our solution. We propose a novel adversarial training technique Ruler particularly designed for improving fairness. We decompose the training procedure into two phases: warm-up and discriminative and iterative adversarial training. In the first phase, Ruler follows the normal training to construct a model such that the normal functionalities (i.e., model accuracy) are guaranteed. We leverage a validation set to determine whether a preset criterion is met. If so, Ruler moves to the next phase for fairness adversarial training. Unlike traditional adversarial training that treats each input feature indiscriminately, Ruler distinguishes sensitive and non-sensitive attributes. As we aim to expose any potential unfair problems in the model for sensitive attributes, Ruler does not constrain the perturbation on these input features. That is, as long as the adversarial perturbations are within the valid value range, they are legitimate. For non-sensitive attributes, Ruler uses a small L_∞ bound to ensure that the generated sample is within the neighborhood. While classic adversarial training induces non-trivial accuracy degradation, we address the problem by limiting the number of discriminatory instances in each training batch. Ruler also introduces a queue to ensure that all the training samples are utilized in each epoch. This is important as there are a limited number of training samples (for representing the whole input distribution). Missing any of those training samples may affect the final accuracy. The blue lines in Figure 1 (Acc_Ruler denotes the accuracy and Unfair_Ruler the unfairness) show the results of Ruler on the Adult dataset. Observe that Ruler achieves a higher accuracy (83.73%) than PGD adversarial training (76.56%) with a similar unfairness value. In Section 5.3, we empirically demonstrate that Ruler outperforms three state-of-the-art techniques in improving fairness.

4 DESIGN

Figure 2 illustrates the overview of Ruler. Given a set of training data, Ruler decomposes the training process into two phases: warm-up and fairness adversarial training. In the first phase, it follows the traditional DNN training procedure that updates model weights guided by the cross entropy loss. Ruler periodically checks the functionality of model by computing the prediction accuracy on the validation data. If it meets a preset criterion, Ruler proceeds to the second phase, in which an adversarial sample generation method is introduced for generating IDIs (①). Different from existing techniques, Ruler relaxes the conditions in Definition 1 by allowing

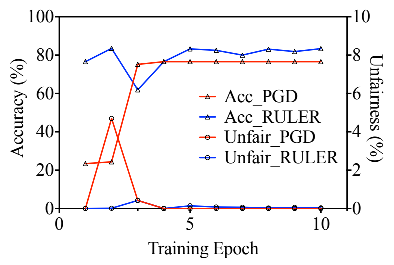


Figure 1: Comparison of PGD adversarial training and RULER.

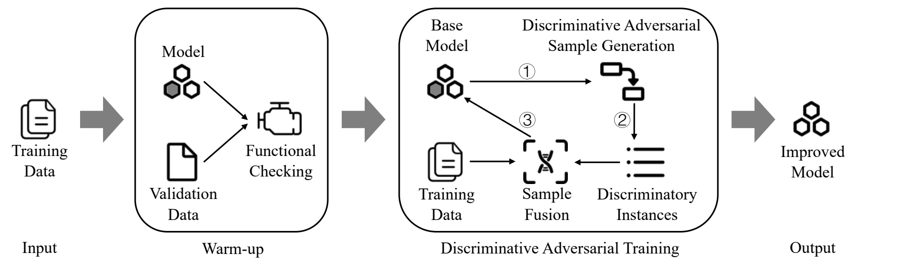


Figure 2: Framework of RULER

small perturbations on non-sensitive attributes without requiring coupling samples. It adopts different constraints on the sensitive and non-sensitive attributes when adversarially perturbing the input (②). Details are discussed in Section 4.2. RULER combines the generated discriminatory samples with normal training data using a sample fusion method (③). This method balances the percentage of normal samples and IDIs in each training batch. All the original training data are guaranteed to be utilized in every training epoch. The training is iterative and terminates when it converges. Please see details in Section 4.2. Note that iterative training is easily affordable in RULER due to the low cost in generating discriminatory samples. In contrast, existing techniques are not iterative. Finally, RULER outputs a model with improved fairness.

4.1 Warm-up

The goal of RULER is to improve model fairness while preserving normal functionalities (i.e., high test accuracy). It is known that adversarial training causes undesirable accuracy degradation on the trained model, which is also demonstrated in Figure 1 in Section 3. In addition, unlike adversarial examples existing in almost all the models, IDIs might not be found when the model has a low accuracy. This is because the existence of IDIs relies on the model overfitting on certain sensitive attribute values, causing the model to misclassify when these attributes are modified. If a model is not well-trained, it may underfit on these attributes and hence is not easy to be exploited with discriminatory cases. To this end, we resort to first constructing a base model with a reasonable accuracy such that we can further explore IDIs regarding this model. The cross entropy loss $\mathcal{L}_{ce}$ is utilized for updating model weights during the warm-up as follows.

$$\arg\min_{\theta} \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \left[\mathcal{L}_{ce}\big(f_\theta(x), y\big)\right], \quad \mathcal{D} = \{\mathcal{X}, \mathcal{Y}\}, \tag{6}$$

where θ denotes the weight parameters of the DNN model $f_\theta(\cdot)$; (x, y) is a sample from the training dataset $\mathcal{D}$ with $x \in \mathcal{X}$ the input data and $y \in \mathcal{Y}$ the output label. The training aims to minimize the expectation of the cross entropy loss over the whole training dataset. To validate the functionality of the model during warm-up, RULER leverages the validation set to compute the prediction accuracy, which is a common practice in DNN training [17, 34, 45]. We use a threshold to determine whether the base model is satisfactory for proceeding to the next training phase.

4.2 Discriminative and Iterative Adversarial Training

To improve model fairness, a commonly employed approach is to generate IDIs and then repair the model by training on these cases. Those instance generation methods aim to explore the input space where the pre-trained model is unfair regrading some sensitive attribute(s). Existing works only focus on the discrimination of the final state of the subject model and aim to repair it regardless the following states (e.g., the state after retraining). They hence can hardly consider the constantly changing states of a model during retraining regarding discrimination. As we will show in Section 5.3, the design choice is likely due to the high cost of generating discriminatory samples in existing works. Just like adversarial examples, discriminatory instances can still be found after one-time repair by existing works (as discussed in Section 3). It is hence crucial to model the changing discriminatory states of the DNN during retraining. We formulate it as a min-max optimization problem in the following.

$$\arg\min_{\theta} \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \left[\max_{x'=g(x)} \mathcal{L}_{ce}\big(f_\theta(x'), y\big)\right], \tag{7}$$

where $g(\cdot)$ is our discriminative adversarial sample generation function that produces IDIs, which is discussed later in this section. The inner maximization problem aims to find an IDI for a given input x that has a maximum cross entropy loss. This is a harder case than simply finding an arbitrary discriminatory instance as in the existing works. The outer minimization problem is to find a set of optimal parameters so that the hard discriminatory cases produced by the inner problem are eliminated. This is the problem of training a fair DNN model against the hardest discriminatory scenarios.

Discriminative Adversarial Sample Generation. Existing generation methods [42, 43, 45] strictly follow Definition 1 to produce IDIs. That is, they mutate both sensitive and non-sensitive attributes by requiring a coupling sample that only differs on sensitive attributes from an input. They also only perturb one single sensitive attribute at a time and do not constrain the change on non-sensitive attributes. This however largely restricts the possibility of exploring the neighborhood of the input, especially when the input space $\mathcal{X}$ is limited and only a subset of the whole distribution is available. Existing techniques have low effectiveness and efficiency as demonstrated in Section 3. We relax the conditions in Definition 1 by allowing any perturbations on non-sensitive attributes within a small bound (no coupling sample is required). For sensitive attributes, there is no constraint on the magnitude of perturbations

Algorithm 1 Discriminative Adversarial Sample Generation

```
Input: input sample $x$, generation budget $K$, DNN model $f$
Output: success flag true/false, discriminatory instance $x'$
1:  $x^0 \leftarrow x$
2:  for $i$ in $0...K$ do
3:      $x^{i+1} \leftarrow x^i + \text{sign}\Big(\nabla_{x^i} \mathcal{L}_{ce}\big(f(x^i), y\big)\Big)$
4:      $x^{i+1} \leftarrow \text{clip}_p(x^{i+1})$          ▷ $p$: sensitive attributes
5:      $x^{i+1} \leftarrow \text{clip}_q(x^{i+1})$          ▷ $q$: non-sensitive attributes
6:      if $f(x) \neq f(x^{i+1})$ then
7:          $x' \leftarrow x^{i+1}$
8:          return true, $x'$
9:      end if
10: end for
11: return false, $x$
```

as long as they are within the valid value range. We use function g in Equation 7 to generate hard discriminatory instances. Formally,

$$\begin{aligned} &\max_{g(x)} \mathcal{L}_{ce}\Big(f_\theta\big(g(x)\big), y\Big), \\ &\text{s.t.} \quad \forall q \in A\backslash P, \ ||g(x)_q - x_q||_\infty \leq \epsilon, \end{aligned} \tag{8}$$

where $q \in A\backslash P$ denotes non-sensitive attributes; $g(x)_q$ is the value of the generated instance for attribute q; ϵ is the upper bound of the allowed perturbation for non-sensitive attributes. Note that we do not constrain the perturbation on sensitive attributes as we aim to explore as many cases as possible when perturbing these attributes. In addition, multiple sensitive attributes can be explored simultaneously as the gradient can be applied on all the attributes. The discriminative adversarial sample generation function g can be instantiated in various ways following Equation 8.

We use Algorithm 1 to realize g. Given an input x, we first compute the gradient with respect to all the attributes using the cross entropy loss (line 3). Note that each input attribute has already been discretized into integer values. We hence directly use the gradient sign as the perturbation without a hyper-parameter α to control the magnitude as in Equation 4. Our discriminative sample generation distinguishes sensitive and non-sensitive attributes, namely, we use different clipping functions to project attributes to their desired value ranges. For sensitive attributes, we clip the perturbed values to the valid range of these attributes (line 4). For non-sensitive attributes, the mutated values shall satisfy the condition in Equation 8, whose magnitude is bounded by ϵ (line 5). We use $\epsilon = 1$ in the paper. If the perturbed sample at some step is already a discriminatory instance, we output this instance and stop the generation process (lines 6-9). Otherwise, we keep mutating the input until the budget K. We return with the original sample if no discriminatory instance can be found (line 11).

Sample Fusion. The key of improving fairness is to incorporate generated IDIs in the training as illustrated in Equation 7. Directly adopting adversarial training in our context leads to non-trivial accuracy degradation as demonstrated in Section 3. We hence decompose the training procedure into two phases where the first phase ensures a reasonable accuracy, which is discussed earlier in this section. However, when training on discriminatory instances, the normal functionality may still be compromised if not designed carefully. Our design hence considers the balance between normal

Algorithm 2 Sample Fusion

```
Input: training set $\mathcal{X}$, batch size $b$, model $f$, threshold $\eta$
Output: repaired model $f'$
1:  $f' \leftarrow f$
2:  $batch \leftarrow \emptyset$                          ▷ batch of samples
3:  $num\_idi \leftarrow 0$        ▷ number of individual discriminatory instances
4:  $Q \leftarrow Queue()$
5:  $Q.EnQueue(\mathcal{X})$
6:  while $Q.Empty() == $ false do
7:      while $batch.size < b$ do
8:          $x \leftarrow Q.DeQueue()$
9:          $batch \leftarrow batch \bigcup \{x\}$
10:         if $batch.size == b$ then
11:             break
12:         end if
13:         $flag, x' \leftarrow DiscriminativeAdvGen(x, K, f')$    ▷ Algorithm 1
14:         if $flag == $ true then
15:             $batch \leftarrow batch \bigcup \{x'\}$
16:             $num\_idi \leftarrow num\_idi + 1$
17:         end if
18:         if $num\_idi \geq \eta \times b$ then
19:             $batch \leftarrow batch \bigcup Q.DeQueue(b - batch.size)$
20:             break
21:         end if
22:     end while
23:     $f' \leftarrow train(f', batch)$          ▷ update $f$ using cross entropy loss
24: end while
```

training samples and generated discriminatory instances. Excessive normal samples leads to limited fairness improvement and too many discriminatory instances causes accuracy degradation. We hence limit the number of discriminatory instances in each training batch using a hyper-parameter η. If the ratio of discriminatory instances exceeds η in a batch, we stop including more such instances in the current batch. In addition, since there are only a limited number of samples in the original training dataset (for representing the whole input distribution), missing any of them during training may lead to low accuracy. We encourage incorporating all original samples in each training epoch. Particularly, we leverage a queue to monitor the usage of samples. A sample is dequeued when used in the training. An epoch of training is finalized when no sample is left in the queue.

Algorithm 2 illustrates the sample fusion procedure in one training epoch. Note that Ruler trains for multiple epochs to obtain a final repaired model. We initialize necessary variables in lines 1-5, such as *batch* for storing a batch of samples, *num_idi* for counting the number of IDIs in the batch, and Q for monitoring the usage of original training samples. For a sample x in the training set, we add it to the current training batch if the batch is not full (lines 8-12). Otherwise, it will be added to the next batch. We then use Algorithm 1 to generate a discriminatory instance x' for the given x in line 13. If the generated sample can induce misclassification, we add this sample x' to the batch as well and increase *num_idi* by one (lines 14-17). When the total number of IDIs exceeds the preset threshold, we fill the remaining batch with normal samples from the training set (lines 18-21). The model parameters are then updated according to the cross entropy loss in line 23. Observe that all the training samples are utilized in one epoch. Multiples rounds

Table 2: Dataset statistics

Dataset	#Attributes	Sen. Att.	Size
Adult	12	age, race, gender	48,842
Statlog	24	age, gender	1,000
Bank	16	age	45,211
COMPAS	12	race	5,278

of Algorithm 2 are applied when repairing a DNN model. RULER outputs a repaired model with improved fairness when the training converges and terminates.

5 EVALUATION

We conduct experiments to answer the following questions:

RQ1. How effective is RULER in improving fairness?
RQ2. How effective is RULER in generating individual discriminatory instances?
RQ3. How efficient is RULER in generating individual discriminatory instances?
RQ4. How does the warm-up phase affect RULER's performance?
RQ5. How do hyper-parameters in RULER affect the results?
RQ6. Why does RULER work with non-sensitive attributes perturbed without coupling samples?

5.1 Experimental Setup

Datasets and Models. We evaluate RULER on four real-world datasets, which are most commonly used in studying individual discrimination problems [2, 16, 19, 37, 41–43], including Adult [22], Statlog [18], Bank Marketing [30], and COMPAS [29]. Details of each dataset are shown in the following:

- **Adult** is a dataset for predicting whether one's income exceeds $50K/yr based on census data. It is also known as Census Income [42, 43] or Adult Income [41] dataset. It contains 12 attributes with three sensitive attributes.
- **Statlog** is a dataset used to classify customers' credit risk level. It is also known as German Credit dataset [41–43]. It has 24 attributes with two sensitive ones (age and gender).
- **Bank Marketing (abbr. Bank)**: is a dataset from direct marketing campaigns (phone calls) of a Portuguese banking institution. It is used to predict whether a client will subscribe a term deposit.
- **COMPAS** is a dataset for assessing the likelihood of a criminal defendant re-offending. It is also known as COMPAS Score [41]. It has 12 attributes with race the sensitive one.

Table 2 presents the statistics of the four datasets, including the number of attributes, sensitive attributes, and the size of the dataset. We use a DNN architecture with 6 fully-connected layers, which is the same as existing works [40, 41, 43].

Settings. For a fair comparison, all the baselines are configured with the settings that have the best performance reported in the original papers [42, 43, 45]. Table 3 shows the hyper-parameters used in our experiment for reproducing baselines. In the table, *num_c* denotes the number of clusters for global generation; *num_g* denotes the number of seeds for global generation; *max_iter* denotes the maximum number of iterations for each seed; *step_size* denotes the perturbation size for each iteration. We also set the maximum

Table 3: Baseline hyper-parameter settings. Global and local denote the corresponding hyper-parameters used in the global and local phases, respectively. Symbol '-' denotes the corresponding hyper-parameter is not required.

Parameter	Value (global/local)		
	ADF [43]	EIDIG [42]	NeuronFair [45]
num_c	4/-	4/-	4/-
num_g	1,000/-	1,000/-	1,000/-
max_iter	40/1,000	10/1,000	10/1,000
step_size	1.0/1.0	1.0/1.0	1.0/1.0

iteration *max_iter* to 10 and the perturbation size of each iteration *step_size* to 1.0 in RULER, which are the same as in [42, 43]. We set the ratio of discriminatory instances in a batch to $\eta = 0.3$ and the number of adversarial training epochs to 70. We study the effect of these two hyper-parameters in Section 5.3.

All the experiments are conducted on a server with Intel Xeon Gold 5218R 2.10GHz CPU, Nvidia RTX 3090, and 503 GB RAM. The operating system is Ubuntu 20.04.

5.2 Evaluation Metrics

We leverage four metrics in the evaluation, including fairness improvement, the total number of generated individual discriminatory instances (#IDIs), the generation diversity, and the time cost.

Fairness Improvement. The purpose of generating IDIs is to improve model fairness regarding sensitive attributes. We adopt an existing unfairness estimation method to evaluate the unfairness of the original model and the repaired model. The estimation method is proposed by Sakshi Udeshi et al. [37] based on the Law of Large Numbers (LLN), which has been extensively used in the literature [37, 42, 43]. Specifically, the method randomly samples a large set of instances and computes the percentage of IDIs in this set. For one trial of evaluation, it first generates m samples uniformly at random, which are independent and identically distributed (IID). It then passes these samples to the model and counts the number of samples being discriminatory, denoted as m'. The estimated unfairness for this trial is hence $\overline{e} = \frac{m'}{m}$. The method carries out T trails to obtain the final estimated unfairness $\overline{E} = \frac{\sum_{i=1}^{T} \overline{e}_i}{T}$. The lower the estimated unfairness, the fairer the model is. We use $m = 10000$ and $T = 100$ in our experiments for estimating unfairness. We then calculate the fairness improvement using the following equation.

$$\frac{|\overline{E}_{repaired} - \overline{E}_{original}|}{\overline{E}_{original}} \times 100\%, \tag{9}$$

where $\overline{E}_{original}$ is the unfairness of the original model and $\overline{E}_{repaired}$ the unfairness of the model repaired by various methods.

Number of Individual Discriminatory Instances (#IDIs). A true IDI shall satisfy the conditions in Definition 1. To evaluate the effectiveness of different instance generation methods, for the generated instances, we count the total number of true IDIs according to Definition 1. Note that duplicate instances are filtered out.

Generation Diversity. The generation diversity evaluates how diverse our generated instances are compared to baselines. The more diverse the generated samples are, the more discriminatory space

Table 4: Overall performance on fairness improvement

Dataset	Sen. Attr.	Original	ADF		EIDIG		NeuronFair		RULER	
			Repaired	Improve.	Repaired	Improve.	Repaired	Improve.	Repaired	Improve.
Adult	Age	11.78%	4.45%	62.22%	2.28%	80.65%	3.80%	67.74%	**0.50%**	**95.76%**
	Race	9.33%	2.61%	72.03%	1.00%	89.28%	5.15%	44.80%	**0.29%**	**96.89%**
	Gender	3.81%	1.24%	67.45%	0.89%	76.64%	2.81%	26.25%	**0.07%**	**98.16%**
Statlog	Age	30.80%	1.80%	94.16%	2.20%	92.86%	6.44%	79.09%	**0.00%**	**100.00%**
	Gender	9.20%	3.50%	61.96%	2.40%	73.91%	4.70%	48.91%	**0.00%**	**100.00%**
Bank	Age	10.36%	3.73%	64.00%	2.83%	72.68%	**0.00%**	**100.00%**	0.60%	94.21%
COMPAS	Race	0.05%	0.07%	-40.00%	0.04%	20.00%	0.01%	84.40%	**0.00%**	**100.00%**
Average		10.76%	2.49%	54.55%	1.66%	72.29%	3.27%	64.45%	**0.21%**	**97.86%**

Table 5: Model accuracy before/after repair

Dataset	Original	Repaired			
		ADF	EIDIG	NeuronFair	RULER
Adult	84.32%	83.66%	84.12%	84.29%	83.82%
Statlog	78.00%	77.25%	77.00%	76.00%	77.50%
Bank	89.22%	89.03%	89.14%	87.93%	89.12%
COMPAS	76.61%	75.28%	74.72%	70.73%	75.50%
Average	82.04%	81.31%	81.25%	79.73%	**81.49%**

the technique explores. We use the following metric to calculate the diversity as in [45].

$$GD_{\text{RULER}}(\rho_{cos}, B) = \frac{CR_{\text{RULER}-B}}{CR_{B-\text{RULER}}}$$

$$CR_{\text{RULER}-B} = \frac{\#\text{IDIs of baselines fall in } \prod_{\text{RULER}}}{\#\text{IDIs of baselines}} \tag{10}$$

$$CR_{B-\text{RULER}} = \frac{\#\text{IDIs of RULER fall in } \prod_{B}}{\#\text{IDIs of RULER}},$$

where B denotes the baseline method; $CR_{\text{RULER}-B}$ represents the coverage rate of RULER's IDIs to baseline's IDIs; $\prod_{\text{RULER}}$ is the area with RULER's IDIs as the center and cosine distance ρ_{cos} as the radius. The RULER's IDIs are more diverse when $GD_{\text{RULER}} > 1$.

Efficiency. We compute the time cost of generating 1,000 IDIs (#sec/1,000 IDIs) following [43, 45].

5.3 Evaluation Results

In this section, we present and analyze the experimental results to answer the five research questions.

RQ1: How effective is RULER in improving fairness?

Table 4 and Table 5 show the unfairness and accuracy of repaired models by different techniques. In Table 4, the first two columns denote the evaluated dataset and sensitive attribute. The third column shows the unfairness of the original model. Column Repaired presents the unfairness of repaired models and column Improvement the relative improvement over the original model (computed using Equation 9). Observe that existing techniques have reasonable performance in improving DNN fairness. However, they fail in some cases. For instance, ADF cannot improve the fairness for Race attribute on COMPAS dataset with -40% improvement. EIDIG has slightly better performance with 20% improvement. NeuronFair can further improve the result (with the sacrifice of accuracy as shown in Table 5) but still has discriminatory cases. RULER, on the other side, can completely remove the discrimination against Race on COMPAS. The COMPAS dataset is to assess the likelihood of a criminal defendant re-offending. The decision made by DNNs for this task is critical as any discrimination against Race can cause negative social impacts. RULER is able to completely address the problem. For the Statlog dataset, ADF, EIDIG, and NeuronFair have limited improvement for the Gender attribute, i.e., from 48.91% to 73.91%. RULER can reduce the unfairness to 0% (with 100% improvement). Overall, RULER has the largest fairness improvement with 97.86%, outperforming the three state-of-the-art techniques. Table 5 shows the model accuracy before and after repair. Observe that

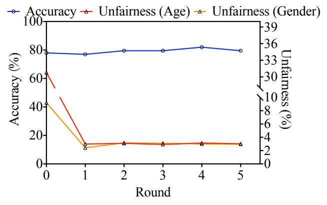


Figure 3: Iterative training using EIDIG

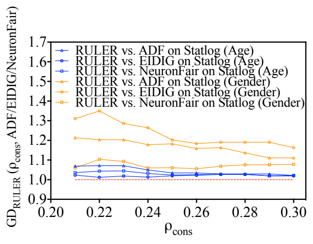


Figure 4: Diversity of generated IDIs

almost all the repaired models have minimal accuracy degradation except for NeuronFair on COMPAS. The repaired model by NeuronFair on COMPAS has more than 5% accuracy degradation, which is substantial. We suspect this is due to incorrect identifications of biased neurons.

As RULER iteratively generates individual discriminatory instances (IDIs) and uses them for repair, we also study the performance of iteratively applying existing techniques. We use one of the state-of-the-art techniques EIDIG and the Statlog dataset for the study. We run EIDIG for five rounds. In each round, EIDIG generates IDIs and repair the previous round model following the procedure in the original paper [42]. Figure 3 shows the results of EIDIG for the different rounds. The x-axis denotes the training round and the y-axes denote the accuracy on the left and the unfairness on the right. Observe that EIDIG can improve the fairness in the first round but it saturates after that. This is because EIDIG and other existing methods generate IDIs for each sensitive attribute independently. They do not consider the correlations between different sensitive attributes. Once they fix the independent discrimination problem for a single attribute, they can hardly further improve the overall fairness. In addition, EIDIG and existing techniques use majority voting to assign labels for the generated samples. The first-round model may already learn the pattern of those samples from majority voting, which is fixed in later rounds. Table 6 shows the cost of each part in each round. Column Global and Local denote the two generation phases in EIDIG that focus on different parts of the input. Column Retrain denotes retraining the model using the original training samples and IDIs. Observe that EIDIG takes more than 11 hours to repair a model for the first round. As the first-round repaired model has better fairness, the number of generated IDIs decreases in later rounds so as the repair time. However, it still takes more than 1.5 hours. In total, it takes 18.73 hours to run EIDIG for 5 rounds, where RULER only needs 6 minutes to train for

Table 6: Time cost (in seconds) of iteratively applying EIDIG

Round	IDI Generation				Retrain	Total
	Global		Local			
	Age	Gender	Age	Gender		
1	286	26,111	495	15,586	106	42,834
2	471	2,268	536	1,522	10	5,279
3	467	2,091	586	2,341	14	5,893
4	705	2,634	702	2,097	12	6,856
5	708	2,387	694	2,061	11	6,569
Total	2,620	35,420	3,013	23,607	152	67,431

Table 7: Number of individual discriminatory instances (#IDIs) generated in 300 seconds

Dataset	Sen. Att.	ADF	EIDIG	NeuronFair	Ruler
Adult	Age	540	7,699	6,817	**26,342**
	Race	164	281	6,137	**24,206**
	Gender	215	4,477	3,807	**22,610**
Statlog	Age	2,240	7,847	1,388	**33,843**
	Gender	570	2,894	1,729	**27,404**
Bank	Age	1,524	9,699	3,918	**27,813**
COMPAS	Race	103	3,389	952	**11,324**
Average		876	5,184	3,535	**24,792**

Table 8: Time cost (in seconds) of generating 1,000 IDIs

Dataset	Sen. Att.	ADF	EIDIG	NeuronFair	Ruler
Adult	Age	218.31	163.96	105.64	**11.97**
	Race	349.78	300.61	114.25	**13.25**
	Gender	237.66	178.20	121.56	**10.91**
Statlog	Age	142.32	117.29	103.91	**14.45**
	Gender	220.00	180.35	296.46	**17.04**
Bank	Age	156.34	118.49	116.52	**8.04**
COMPAS	Race	236.41	198.57	187.50	**26.31**
Average		222.97	179.64	149.41	**14.57**

Table 9: Effect of warm-up. Ruler-NT denotes performing discriminative adversarial training from scratch (without warm-up).

Dataset	Sen. Attr.	Original		Ruler-NT		Ruler	
		Accuracy	Unfairness	Accuracy	Unfairness	Accuracy	Unfairness
Adult	Age	84.32%	11.78%	83.34%	0.33%	83.82%	0.50%
	Race	84.32%	9.33%	83.34%	0.09%	83.82%	0.29%
	Gender	84.32%	3.81%	83.34%	0.92%	83.82%	0.07%
Statlog	Age	78.00%	30.80%	70.50%	0.00%	77.50%	0.00%
	Gender	78.00%	9.20%	70.50%	0.00%	77.50%	0.00%
Bank	Age	89.22%	10.36%	89.05%	0.80%	89.12%	0.60%
COMPAS	Race	76.61%	0.05%	72.35%	0.00%	75.50%	0.00%

70 epochs. In summary, iteratively applying existing techniques is less effective and efficient than Ruler.

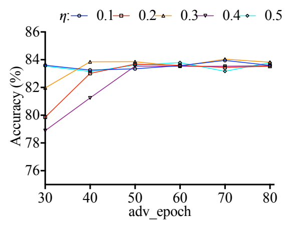
RQ2: How effective is Ruler in generating individual discriminatory instances?

We consider two aspects of instance generation, namely, the number of generated individual discriminatory instances (#IDIs) and their diversity. Table 7 shows #IDIs generated by different methods in 300 seconds. Observe that Ruler substantially outperforms baselines with 7-28 times more generated IDIs. This allows Ruler to explore more discrimination problems resident in the model. For studying the diversity, we randomly select a few datasets and sensitive attributes. Figure 4 presents the diversity of generated IDIs, which is computed according to Equation 10. It measures how different the generated IDIs by different generation methods are. The x-axis denotes the radius and y-axis the diversity value. The diversity value larger than 1 indicates that Ruler has more diverse IDIs than the compared counterpart. Observe that the diversity values are all larger than 1 in Figure 4. Particularly, Ruler has relatively more diverse IDIs compared to ADF and EIDIG on Statlog for Gender. With the increase of radius, the diversity values decrease. This is reasonable as a large radius may already cover all the generated IDIs.

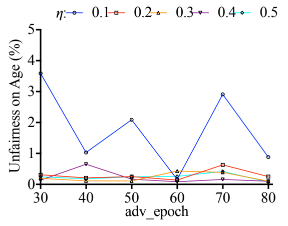
RQ3: How efficient is Ruler in generating individual discriminatory instances?

Table 8 shows the time cost of generating one thousand IDIs by different techniques. Observe that ADF has the highest time cost with an average of 222.97 seconds. EIDIG has lower time cost of 179.64 seconds on average. NeuronFair further reduces the time cost. Ruler is the most efficient method with 14.57 seconds for generating 1,000 IDIs, which is 10-15 times faster than baselines.

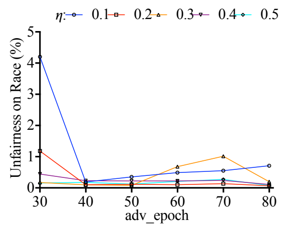
RQ4: How does the warm-up phase affect Ruler's performance?



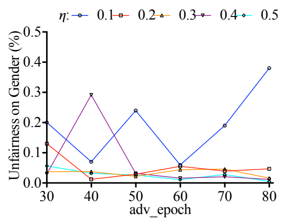
(a) Effect on accuracy



(b) Effect on unfairness for Age



(c) Effect on unfairness for Race



(d) Effect on unfairness for Gender

Figure 5: Effect of hyper-parameters η and *adv_epoch*

In order to preserve normal functionalities of the DNN model, we introduce a warm-up phase in Ruler. We conduct an ablation study to understand the effect of this phase on the accuracy and unfairness of repaired models. Table 9 presents the results. Column Ruler-NT denotes the results without the warm-up. Observe that Ruler obtains better accuracy on repaired models compared to Ruler-NT. Especially on Statlog and COMPAS, the repaired models by Ruler have 7% and 3.15% higher accuracy than those of Ruler-NT, respectively. The fairness improvement are similar for Ruler and Ruler-NT, delineating the effectiveness of our discriminative and iterative adversarial training in improving DNN fairness.

RQ5: How do hyper-parameters in Ruler affect the results?

Two hyper-parameters are used in Ruler, namely, η and *adv_epoch*, to control the number of IDIs in each batch and the number of adversarial training epochs, respectively. We conduct an ablation study on Adult dataset and the results are shown in Figure 5. From Figure 5(a), observe that with the increase of epochs, the accuracy increases as the model gradually converges. Different values of η have slightly different convergence speeds but they all converge at 50 epochs and have similar accuracy. We use *adv_epoch* = 70 in all our experiments. Figures 5(b), 5(c), and 5(d) show the results on unfairness for different sensitive attributes. Parameter η = 0.1 has

Table 10: Description of attributes in COMPAS

Attribute	Description
sex	0: "Female"; 1: "Male"
age	0: "25-45"; 1: "Greater than 45"; 2: "Less than 25"
race	**0: "African-American"; 1: "Caucasian"**
juv_fel_count (JFC)	a continuous variable containing the number of juvenile felonies
juv_misd_count (JMC)	a continuous variable containing the number of juvenile misdemeanors
juv_other_count (JOC)	a continuous variable containing the number of prior juvenile convictions that are not considered either felonies or misdemeanors
priors_count (PC)	a continuous variable containing the number of prior crimes committed
days_b_screening_arrest (DBSA)	days between the arrest and COMPAS screening (a negative value means the screening is taken before the arrest)
jail_time (JT)	the total number of days arrested in jail
date_dif_in_jail (DDIJ)	the date interval between two times arrested in jail
charge_degree (CD)	0: "Felony"; 1: "Misdemeanor charge"
is_recid (IR)	0: "not a recidivist"; 1: "a recidivist"

the largest impact on unfairness for Age and Gender. The unfairness results fluctuate during training. The is because the number of IDIs is too small, which makes the model not able to fully learn IDIs in each epoch. Other values of η have minimal impact on unfairness for all three sensitive attributes. We use $\eta = 0.3$ in the paper.

RQ6: Why does Ruler work with non-sensitive attributes perturbed without coupling samples?

As Ruler perturbs both sensitive and non-sensitive attributes simultaneously without requiring coupling samples, we measure the number of perturbed non-sensitive features in IDIs. On average, 4.62/9, 9.85/22, 7.23/15, and 5.69/11 non-sensitive features are perturbed for Adult, Statlog, Bank, and COMPAS, respectively. As Ruler uses the original labels for those IDIs during model repair, one may wonder whether this is reasonable to preserve the labels for those IDIs with such a number of changes on the input features. We study a few cases in the COMPAS dataset as an example to show the validity of label preserving.

The COMPAS dataset is to assess the likelihood of a criminal defendant re-offending (label 1 means high risk and label 0 means low risk). It has 12 attributes with **race the sensitive one**. Table 10 lists each attribute name and its meaning. Table 11 shows a few example pairs of original samples and our generated adversarial samples. The first row denotes the attributes. The following every two rows show the original sample (Original) and the generated adversarial sample (Adversarial). The last column presents the predicted label of the sample.

In the first case, six attributes are perturbed, namely, age, race, priors_count (PC), days_b_screening_arrest (DBSA), jail_time (JT), and date_dif_in_jail (DDIJ), respectively. Observe that the sensitive attribute race is changed from "Caucasian" to "African-American". Other attributes in the adversarial example are similar to those in the original sample. Values for priors_count (PC) and jail_time (JT) of the adversarial sample are smaller than those of the original sample. This adversarial sample is predicted as label 1 by the model, meaning the defendant has high risk, which does not seem reasonable compared to the original sample. We hence use label 0 during training.

Table 11: Case study

Attribute	sex	age	race	JFC	JMC	JOC	PC	DBSA	JT	DDIJ	CD	IR	Label
Original_1	1	0	1	0	0	0	5	0	6	7	0	0	0
Adversarial_1	1	1	0	0	0	0	4	1	5	6	0	0	1
Original_2	1	0	0	0	0	0	0	-1	41	41	0	0	0
Adversarial_2	0	0	1	0	0	0	0	-2	40	42	0	0	1

In the second case, the perturbed attributes are sex, race, days_b _screening_arrest (DBSA), jail_time (JT), and date_dif_in_jail (DDIJ), respectively. Observe that the race attribute is changed from "African-American" to "Caucasian". The value of jail_time (JT) is reduced. The date interval (date_dif_in_jail) is larger between two times arrested in jail. The predicted label for this adversarial sample however is 1 (high risk), different from the original label. We hence use label 0 for training.

The above two cases demonstrate that although Ruler perturbs non-sensitive attributes, it ensures that the generated sample is within the neighborhood of the original input as we use a small bound on perturbed attributes. Samples within the neighborhood of the original input hence shall maintain the same label, which avoids introducing abnormal data distribution.

6 THREATS TO VALIDITY

Limited Model Structures and Machine Learning Algorithms. We evaluate on the fully-connected deep neural networks in the experiments following the literature [42, 43, 45] as the tasks are relatively simple. The key idea of Ruler is generic and can be directly applied on more complex neural networks such as convolutional neural networks (CNNs) as long as the model is differentiable. If non-NN machine learning (ML) algorithms are not differentiable, there are many existing techniques such as evolution algorithm, gradient approximation [5], etc., to obtain the perturbation directions. Ruler can then constrain the obtained perturbation according to Equation 8 and repair those algorithms. For ML models with primitive inputs such as images, the key challenge is to identify sensitive regions of the input regarding fairness. One possible solution is to first use the gradient (e.g. Grad-CAM [31]) to locate such regions and then apply Ruler to improve the fairness. For NLP tasks, it is possible to construct a list of sensitive words and Ruler can leverage the gradient to search for the replacement of those words. The fairness then can be improved by Ruler's adversarial training. We leave the experimental exploration to future work.

Access to Model Parameters. Ruler is a white-box technique that generates individual discriminatory instances based on the gradient according to the loss function of the subject model. It requires the full access to the model parameters, which is the same as in the literature [43, 45]. It is a common practice to have the full knowledge of the subject model for repairing [45].

Hyper-parameter. The step size of Ruler during instance generation is set to 1 following existing work [43]. The datasets evaluated in the paper have categorized attributes. It hence is straightforward to use 1 as the step size. For other non-categorical datasets, further study may be needed to determine the optimal step size. We leave the exploration to future work.

7 RELATED WORK

There is a line of work aiming to generate test inputs for exposing individual discrimination resident in the model. THEMIS [16] automatically generates individual discriminatory instances (IDIs) by randomly selecting values from the valid range for all the attributes. It measures the discrimination of the subject system based on whether the system discriminates certain attributes. AEQUITAS [37] adopts a two-phase generation method. In the global phase, it generates a random sample in the input domain and exhaustively enumerates all the values for sensitive attributes to obtain an IDIs. In the local phase, it then perturbs other attributes for the generated IDIs in the first phase. However, AEQUITAS can only search in a narrow input space and it easily falls into the local optimum [45]. To improve the diversity of IDIs, SG [2] leverages dynamic symbolic execution to generate IDIs. The aforementioned methods, namely, THEMIS, AEQUITAS, and SG, are mainly designed for traditional machine learning models, such as logistic regression, support vector machine, and decision tree, etc. They may not be applicable to deep neural networks (DNNs) or are very expensive as DNNs are more complex. A few methods have been proposed recently that are specifically designed for DNNs. Zhang et al. [43] propose a gradient-based method called ADF for generating individual discriminatory instances. They demonstrate that ADF can improve the effectiveness and efficiency of IDIs generation for DNNs based on the guidance of gradients. EIDIG [42] extends ADF by leveraging prior gradient information (i.e., momentum) to accelerate the convergence of instance generation. However, it may still suffer from gradient vanishing, leading to local optimization [45]. FairNeuron [17] detects neurons that particularly contribute to sensitive attributes to generate IDIs. Similarly, NeuronFair [45] searches for biased neurons and generates samples to enlarge the activation values of those neurons. More related works can be found in survey [6]. We compare Ruler with three state-of-the-art techniques, namely, ADF, EIDIG, and NeuronFair, in Section 5.3. Ruler outperforms these baselines in improving fairness on four evaluated datasets.

Adversarial training is one of the most effective methods in improving model robustness [23, 28, 32, 36, 39]. They can be leveraged to improve fairness as demonstrated in Section 3. However, it is known that adversarial training induces non-trivial accuracy degradation. We also have the same observation in our context. Ruler improves traditional adversarial training by decomposing the training into two phases and introducing sample fusion to have a better accuracy-fairness balance.

8 CONCLUSION

We propose a novel discriminative and iterative adversarial training method particularly designed for improving DNN fairness. We evaluate our prototype Ruler on four popular datasets. It has 97.86% relative fairness improvement with only 0.55% accuracy degradation on average, outperforming three state-of-the-art techniques.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their constructive comments. This research was supported, in part by IARPA TrojAI W911NF-19-S-0012, NSF 1901242 and 1910300, ONR N000141712045, N000141410468 and N000141712947, DARPA VSPELLS HR001120S0058, and National Natural Science Foundation of China (61932012 and 62141215). Any opinions, findings, and conclusions in this paper are those of the authors only and do not necessarily reflect the views of our sponsors.

REFERENCES

[1] Aniya Aggarwal, Pranay Lohia, Seema Nagar, Kuntal Dey, and Diptikalyan Saha. 2018. Automated Test Generation to Detect Individual Discrimination in AI Models. *CoRR* abs/1809.03260 (2018).
[2] Aniya Aggarwal, Pranay Lohia, Seema Nagar, Kuntal Dey, and Diptikalyan Saha. 2019. Black box fairness testing of machine learning models. In *Proceedings of the 13th Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM, Tallinn, Estonia, 625–635.
[3] Naveed Akhtar and Ajmal S. Mian. 2018. Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey. *IEEE Access* 6, 1 (2018), 14410–14430.
[4] Emily Black, Samuel Yeom, and Matt Fredrikson. 2020. FlipTest: fairness testing via optimal transport. In *Proceedings of the 3th Conference on Fairness, Accountability, and Transparency*. ACM, Barcelona, Spain, 111–121.
[5] Jianbo Chen, Michael I Jordan, and Martin J Wainwright. 2020. Hopskipjumpattack: A query-efficient decision-based attack. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1277–1294.
[6] Zhenpeng Chen, Jie M Zhang, Max Hort, Federica Sarro, and Mark Harman. 2022. Fairness Testing: A Comprehensive Survey and Analysis of Trends. *arXiv e-prints* (2022), arXiv–2207.
[7] Yaya Cheng, Xiaosu Zhu, Qilong Zhang, Lianli Gao, and Jingkuan Song. 2021. Fast Gradient Non-sign Methods. *CoRR* abs/2110.12734 (2021).
[8] Corinna Cortes and Vladimir Vapnik. 1995. Support-Vector Networks. *Machine Learning* 20, 3 (1995), 273–297.
[9] The decision tree classifier: Design and potential. 1977. Philip H Swain and Hans Hauska. *IEEE Transactions on Geoscience Electronics* 15, 3 (1977), 142–147.
[10] Jiankang Deng, Jia Guo, Jing Yang, Alexandros Lattas, and Stefanos Zafeiriou. 2021. Variational prototype learning for deep face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11906–11915.
[11] Li Deng and John C. Platt. 2014. Ensemble deep learning for speech recognition. In *Proceedings of the 15th Annual Conference of the International Speech Communication Association*. ISCA, Singapore, 1915–1919.
[12] Yao Deng, Tiehua Zhang, Guannan Lou, Xi Zheng, Jiong Jin, and Qing-Long Han. 2021. Deep Learning-Based Autonomous Driving Systems: A Survey of Attacks and Defenses. *IEEE Transactions on Industrial Informatics* 17, 12 (2021), 7897–7912.
[13] Yunfeng Diao, Tianjia Shao, Yongliang Yang, Kun Zhou, and He Wang. 2021. BASAR: Black-Box Attack on Skeletal Action Recognition. In *Proceedings of the 31st Conference on Computer Vision and Pattern Recognition*. Computer Vision Foundation / IEEE, virtual, 7597–7607.
[14] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard S. Zemel. 2012. Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*. ACM, Cambridge, MA, USA, 214–226.
[15] Michael Feldman, Sorelle A. Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. 2015. Certifying and Removing Disparate Impact. In *Proceedings of the 21th International Conference on Knowledge Discovery and Data Mining*. ACM, Sydney, NSW, Australia, 259–268.
[16] Sainyam Galhotra, Yuriy Brun, and Alexandra Meliou. 2017. Fairness testing: testing software for discrimination. In *Proceedings of the 11th Joint Meeting on Foundations of Software Engineering*. ACM, Paderborn, Germany, 498–510.
[17] Xuanqi Gao, Juan Zhai, Shiqing Ma, Chao Shen, Yufei Chen, and Qian Wang. 2022. Improving Deep Neural Network Fairness with Adversary Games on Selective Neurons. In *Proceedings of the 44rd International Conference on Software Engineering*. IEEE/ACM, Pittsburgh, PA, USA, to be appear.
[18] Hans Hofmann. 1994. Statlog (German Credit Data) Data Set. https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data). Accessed Mar, 2022.
[19] Max Hort, Jie M. Zhang, Federica Sarro, and Mark Harman. 2021. Fairea: a model behaviour mutation approach to benchmarking bias mitigation methods. In *Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM, Athens, Greece, 994–1006.
[20] Sandy H. Huang, Nicolas Papernot, Ian J. Goodfellow, Yan Duan, and Pieter Abbeel. 2017. Adversarial Attacks on Neural Network Policies. In *Proceedings of the 5th International Conference on Learning Representations – Workshop*. OpenReview.net, Toulon, France, 1–10.
[21] Shuai Jia, Yibing Song, Chao Ma, and Xiaokang Yang. 2021. IoU Attack: Towards Temporally Coherent Black-Box Adversarial Attack for Visual Object Tracking. In *Proceedings of the 31st Conference on Computer Vision and Pattern Recognition*. Computer Vision Foundation / IEEE, virtual.
[22] Ronny Kohavi and Barry Becker. 1996. Adult Data Set. https://archive.ics.uci.edu/ml/datasets/adult. Accessed Mar, 2022.

[23] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2017. Adversarial machine learning at scale. In *ICLR*.

[24] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. 2017. Adversarial examples in the physical world. In *Proceedings of the 5th International Conference on Learning Representations – Workshop (ICLR 2017)*. OpenReview.net, Toulon, France, 1–14.

[25] Louisa Lam and Ching Y. Suen. 1997. Application of majority voting to pattern recognition: an analysis of its behavior and performance. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 27, 5 (1997), 553–568.

[26] Michael P LaValley. 2008. Logistic regression. *Circulation* 117, 18 (2008), 2395–2399.

[27] Jiadong Lin, Chuanbiao Song, Kun He, Liwei Wang, and John E. Hopcroft. 2020. Nesterov Accelerated Gradient and Scale Invariance for Adversarial Attacks. In *Proceedings of the 8th International Conference on Learning Representations*. OpenReview.net, Addis Ababa, Ethiopia, 1–12.

[28] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *Proceedings of the 6th International Conference on Learning Representations-Poster*. OpenReview.net, Vancouver, BC, Canada, 1–28.

[29] ProPublica. 2017. COMPAS Recidivism Risk Score Data and Analysis. https://www.propublica.org/datastore/dataset/compas-recidivism-risk-score-data-and-analysis. Accessed Mar, 2022.

[30] P. Cortez S. Moro and P. Rita. 2012. Bank Marketing Data Set. https://archive.ics.uci.edu/ml/datasets/bank+marketing. Accessed Mar, 2022.

[31] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*. 618–626.

[32] Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. 2019. Adversarial Training for Free! *NeurIPS* (2019).

[33] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *Proceedings of the 2nd International Conference on Learning Representations-Poster*. Banff, AB, Canada, 1–10.

[34] Guanhong Tao, Shiqing Ma, Yingqi Liu, Qiuling Xu, and Xiangyu Zhang. 2020. Trader: Trace divergence analysis and embedding regulation for debugging recurrent neural networks. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*. IEEE, 986–998.

[35] Guanhong Tao, Weisong Sun, and Tingxu Han. 2022. RULER. site: https://github.com/wssun/RULER. Accessed August, 2022.

[36] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. 2018. Ensemble Adversarial Training: Attacks and Defenses. In *ICLR*.

[37] Sakshi Udeshi, Pryanshu Arora, and Sudipta Chattopadhyay. 2018. Automated directed fairness testing. In *Proceedings of the 33rd International Conference on Automated Software Engineering*. ACM, Montpellier, France, 98–108.

[38] Zhipeng Wei, Jingjing Chen, Xingxing Wei, Linxi Jiang, Tat-Seng Chua, Fengfeng Zhou, and Yu-Gang Jiang. 2020. Heuristic Black-Box Adversarial Attacks on Video Recognition Models. In *Proceedings of the the 34th Conference on Artificial Intelligence*. AAAI Press, New York, NY, USA, 12338–12345.

[39] Eric Wong, Leslie Rice, and J Zico Kolter. 2020. Fast is better than free: Revisiting adversarial training. In *ICLR*.

[40] Han Xu, Yao Ma, Haochen Liu, Debayan Deb, Hui Liu, Jiliang Tang, and Anil K. Jain. 2020. Adversarial Attacks and Defenses in Images, Graphs and Text: A Review. *International Journal of Automation and Computing* 17, 2 (2020), 151–178.

[41] Jie M. Zhang and Mark Harman. 2021. "Ignorance and Prejudice" in Software Fairness. In *Proceedings of the 43rd International Conference on Software Engineering*. IEEE, Madrid, Spain, 1436–1447.

[42] Lingfeng Zhang, Yueling Zhang, and Min Zhang. 2021. Efficient white-box fairness testing through gradient search. In *Proceedings of the 30th International Symposium on Software Testing and Analysis*. ACM, Virtual Event, Denmark, 103–114.

[43] Peixin Zhang, Jingyi Wang, Jun Sun, Guoliang Dong, Xinyu Wang, Xingen Wang, Jin Song Dong, and Ting Dai. 2020. White-box fairness testing through adversarial sampling. In *Proceedings of the 42nd International Conference on Software Engineering*. ACM, Seoul, South Korea, 949–960.

[44] Baolin Zheng, Peipei Jiang, Qian Wang, Qi Li, Chao Shen, Cong Wang, Yunjie Ge, Qingyang Teng, and Shenyi Zhang. 2021. Black-box Adversarial Attacks on Commercial Speech Platforms with Minimal Information. In *Proceedings of the 28th Conference on Computer and Communications Security*. ACM, Virtual Event, Republic of Korea, 86–107.

[45] Haibin Zheng, Chen Zhiqing, Du Tianyu, Zhang Xuhong, Cheng Yao, Ji Shouling, Wang Jingyi, Yu Yue, and Chen Jinyin. 2022. NeuronFair: Interpretable White-Box Fairness Testing through Biased Neuron Identification. In *Proceedings of the 44rd International Conference on Software Engineering*. IEEE/ACM, Pittsburgh, PA, USA, to be appear.