Dissecting Global Search: A Simple yet Effective Method to Boost Individual Discrimination Testing and Repair

Lili Quan*, Tianlin Li†, Xiaofei Xie‡, Zhenpeng Chen†, Sen Chen§, Lingxiao Jiang‡, Xiaohong Li*
*Tianjin University, Tianjin, China †Nanyang Technological University, Singapore
‡Singapore Management University, Singapore §Nankai University, Tianjin, China
quanlili@tju.edu.cn, tianlin001@e.ntu.edu.sg, xfxie@smu.edu.sg, zhenpeng.chen@ntu.edu.sg,
tigersenchen@163.com, lxjiang@smu.edu.sg, xiaohongli@tju.edu.cn

2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE) | 979-8-3315-0569-1/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICSE55347.2025.00235

***Abstract*—Deep Learning (DL) has achieved significant success in socially critical decision-making applications but often exhibits unfair behaviors, raising social concerns. Among these unfair behaviors, individual discrimination—examining inequalities between instance pairs with identical profiles differing only in sensitive attributes such as gender, race, and age—is extremely socially impactful. Existing methods have made significant and commendable efforts in testing individual discrimination before deployment. However, their efficiency and effectiveness remain limited, particularly when evaluating relatively fairer models. It remains unclear which phase of the existing testing framework (global or local) is the primary bottleneck limiting performance.**

Facing the above issues, we first identify that enhancing the global phase consistently improves overall testing effectiveness compared to enhancing the local phase. This motivates us to propose Genetic-Random Fairness Testing (GRFT), an effective and efficient method. In the global phase, we use a genetic algorithm to guide the search for more global discriminatory instances. In the local phase, we apply a light random search to explore the neighbors of these instances, avoiding time-consuming computations. Additionally, based on the fitness score, we also propose a straightforward yet effective repair approach. For a thorough evaluation, we conduct extensive experiments involving 6 testing methods, 5 datasets, 261 models (including 5 naively trained, 64 repaired, and 192 quantized for on-device deployment), and sixteen combinations of sensitive attributes, showing the superior performance of GRFT and our repair method.

***Index Terms*—Individual Discrimination, Fairness, DNNs**

I. Introduction

Deep Learning (DL) has achieved remarkable success and demonstrated great potential in tackling complex tasks, particularly in socially critical decision-making applications [1]–[4]. However, they often exhibit unfair behaviors in sensitive application areas [5]–[9], raising significant social concerns and violating the fairness requirements of software [10], [11]. For example, studies [3] show that pedestrian detectors are significantly biased at night, with more females going undetected than males, causing severe security concerns, perpetuating gender discrimination, and exacerbating social inequalities. Therefore, it is crucial to enhance fairness throughout the entire development and deployment process of DL systems, particularly in socially critical scenarios.

Specifically, existing studies on deep neural network fairness can be primarily divided into two categories: individual discrimination [12]–[15] and group fairness [16]–[19]. Individual discrimination requires similar decisions for similar individuals, while group fairness requires equal treatment for different groups based on a protected attribute. Individual discrimination is capable of identifying discriminatory behaviors that may be ignored by group fairness measures. Specifically, individual discrimination allows for a more powerful and granular examination of discriminatory behavior that arises due to changes in only the sensitive attribute in various contexts. In contrast, group fairness measures may fail to detect discrimination when the model treats the same group oppositely in different situations. Thus, in this work, we primarily focus on addressing the issue of individual discrimination in deep neural networks (DNNs).

To thoroughly test individual discrimination in DNNs, existing methods [14], [15], [20]–[25] primarily generate individual discriminatory instances (IDIs) in a two-phase generation framework. Recognizing that the neighbors of IDIs are likely to be discriminatory, they first identify diverse instances as global seeds (the global phase). Then, they iteratively search the neighbors of these global seeds to uncover as many as possible IDIs (the local phase). Specifically, ADF [23] and EIDIG [15] guide the search direction through gradients in both phases. NeuronFair [24] and DICE [25] improve upon previous methods by optimizing gradient guidance using neuron behaviors or information-theoretic characterization of discrimination. ExpGA [14] collects high-quality global seeds through interpretable methods and then uses a Genetic Algorithm to search for more IDIs in the local phase.

Despite the advanced nature of current efforts, their efficiency and effectiveness remain limited, particularly when evaluating fairer models, even with white-box access (*e.g.*, NeuronFair [24] takes 3,324 seconds to generate only 148 IDIs for models improved by Faire [26], as detailed in table II). To understand which phase is the key bottleneck, we conduct

This work was done during Lili Quan's visit to Singapore Management University. Tianlin Li and Xiaohong Li are the corresponding authors.

1558-1225/25/$31.00 ©2025 IEEE
DOI 10.1109/ICSE55347.2025.00235

a preliminary study comparing the impact of enhancing the global versus local phases under the same search budget (Details in Section II-C). Our findings show that enhancing the global phase is significantly more rewarding: increasing global IDIs consistently improves overall testing effectiveness than enhancing the local phase.

Motivated by this, we mainly focus on improving the global search algorithm to enhance the overall effectiveness and efficiency of fairness testing. Considering this factor, we propose Genetic-Random Fairness Testing (GRFT), a straightforward yet effective and efficient fairness testing method. In the global phase, we utilize a genetic algorithm that effectively guides the search process toward generating more global IDIs from a single seed. The fitness function is designed based on the model output differences between the original and corresponding mutated cases. This significant increase in generated global IDIs allows us to adopt an efficient and less sophisticated search strategy in the local phase while maintaining high effectiveness. Thus in the local phase, we perform a light method (*i.e.*, random search) to explore the neighbors of the identified global IDIs, avoiding time-consuming computations. Additionally, our proposed method requires only black-box access to the target model.

The fitness score used in GRFT typically measures the degree of unfairness or bias in a model's predictions. This insight inspires us to improve model fairness by reducing the model's fitness scores. Specifically, we propose a straightforward yet effective repair approach that constructs instance pairs from the original training data and introduces a novel loss function aimed at directly reducing differences in model outputs between these pairs. This new loss function ensures that while the model continues to perform well on the primary task, it also becomes less likely to exhibit biased behavior.

To conduct a thorough investigation, we extend the performance study beyond existing methods that primarily evaluate vanilla models (*i.e.*, naively-trained models before any repair), which can lead to misleading conclusions about the effectiveness of these testing and repair methods. We comprehensively assess the fairness of models across three dimensions: the vanilla model before repair, the repaired model, and the quantized model which indicates performance when deployed on a device.

The extensive experiments reveal that GRFT can find more IDIs more quickly than all baselines across all models and datasets. For example, on average, across all datasets, compared to the best-performing baseline EIDIG, GRFT takes only 0.96% (*i.e.*, 84 seconds) of the time to discover 3.07 times (*i.e.*, 278,344) more IDIs in the models repaired by Faire. Moreover, compared to existing repair methods, our repair method significantly improves the fairness performance of the model. For example, GRFT on average discovers 2,850.6 IDIs in the models repaired by our method, which is a 60.50% reduction compared to the flipping-based retrained models.

Contributions. In summary, this research makes the following contributions:

- This paper highlights that increasing the number of global IDIs improves testing effectiveness more than enhancing the local phase under the same budget.
- We propose GRFT, a novel method that uses genetic algorithms in the global phase and random search in the local phase for more efficient and effective IDI generation.
- We propose a straightforward yet effective repair method by introducing a novel loss function to reduce differences in model outputs between instance pairs.
- We conduct extensive experiments involving 6 testing methods, 261 models (including 5 naively-trained, 64 repaired, and 192 quantized models), 5 datasets, and 16 combinations of sensitive attributes to demonstrate the efficiency and effectiveness of our method.
- We release our testing and repair tool in https://sites.google.com/view/faireness-testing-grft.

II. Background & Preliminary Study

A. Deep Neural Networks

Deep Neural Networks (DNNs), inspired by the neural networks of the human brain, are renowned for their exceptional performance [27]. These networks typically consist of multiple layers of interconnected neurons.

Definition 1: A Deep Neural Network (DNN) f consists of multiple layers $\langle l_0, l_1, \ldots, l_k, l_o \rangle$, where l_0 is the input layer, l_o is the output layer, and $l_1, \ldots, l_k$ are hidden layers. The inputs of each layer are from the outputs of the previous layer.

In this work, we mainly focus on the classifier $f : X \to Y$, where X is a set of inputs and Y is a set of classes. Given an input $x \in X$, we use $f_l(x)$ to represent the internal features extracted by the layer l (*i.e.*, the neuron output values at l).

B. Problem Definition

1) Individual Discrimination: There are various metrics for evaluating the fairness of machine learning models [28]. Among them, individual fairness asserts that similar inputs differing only in protected attributes should not lead to discriminatory outcomes. We adopt the definition of individual discrimination from [23]. Let the attribute set of the dataset be $A = A_1, A_2, \cdots, A_n$, with $P \subset A$ representing protected attributes (*e.g.*, gender, race, age) and NP denoting non-protected attributes. We define a discriminatory instance as follows.

Definition 2 (Discriminatory Instance): $x = a_1, a_2, \cdots, a_n$ is an arbitrary instance in dataset, where a_i represents the value of attribute A_i. We define x as a discriminatory instance of a DNN when there is a $x' = a'_1, a'_2, \cdots, a'_n$ in the instance space that satisfies the following conditions:

- $\exists p \in P, s.t., a_p \neq a'_p$;
- $\forall q \in NP, s.t., a_q = a'_q$;
- $f(x) \neq f(x')$

2) Repairing Individual Discrimination : The repair could be defined as: *given a DNN f that suffers from individual discrimination, we aim to repair the DNN f as a fairer DNN f'.* Given any input x, if we change some values of the

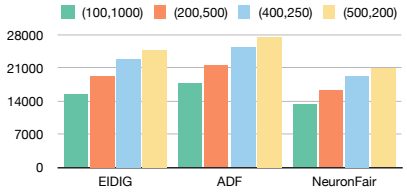

Fig. 1: Result of preliminary study

protected attributes as x', the classification output should be not changed, *i.e.*, $f'(x) = f'(x')$. The definition could be:

$$f^{\theta}(x) = f^{\theta}(x') \wedge f(x) = f^{\theta}(x),$$

where f^{θ} means the new model with the learned parameters θ. For the input x and x', we expect that the decisions of the DNN on these two inputs rely on the features that are as similar as possible, and the original functionality is not affected.

C. Preliminary Study

Existing methods typically generate m IDIs during the global phase and then perform n local iterations for each IDI, resulting in a total of $m \times n$ iterations. We ignore the cost of the global phase here since it is often minimal due to significantly fewer global iterations (*e.g.*, 10 global iterations versus 1000 local iterations). However, it remains unclear: *Given a fixed $m \times n$ budget, how should resources be allocated to maximize testing effectiveness?* To explore it, we investigate the following two strategies:

- *Strategy 1:* Enhance the global phase by increasing m, the number of IDIs, while reducing n, the number of local iterations per IDI.

- *Strategy 2:* Enhance the local phase by giving a smaller m and increasing n, thus focusing on more iterations per IDI.

Existing methods predominantly follow Strategy 2, adopting a lightweight global phase (*e.g.*, generating only one IDI per seed input with 10 iterations) and performing a more intensive local phase (*e.g.*, 1000 iterations per IDI). However, the effectiveness of this approach compared to Strategy 1 remains unclear.

To address this gap, we conducted a preliminary study to compare these two strategies. Using a fixed budget of $m \times n = 100,000$, we evaluated four configurations: (100, 1000), (200, 500), (400, 250), (500, 200), where each pair represents (m, n). As shown in Figure 1, our findings reveal that increasing m (the number of global IDIs) consistently improves testing effectiveness.

III. Methodology

A. Testing Individual Discrimination

Based on findings (in Section II-C) that enhancing the global phase yields greater performance rewards than enhancing the local phase, we designed GRFT, a simple and effective method that combines a more efficient global search algorithm (*i.e.*, Genetic Algorithm) with a lightweight local search algorithm (*i.e.*, Random).

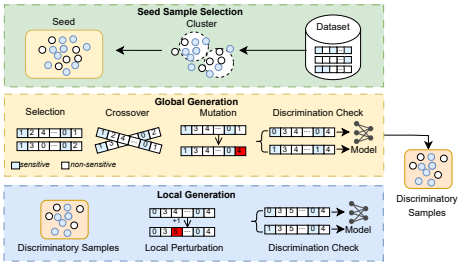

Fig. 2: Overview of GRFT

Algorithm 1: Global Generation

Input : s: a seed sample, max_iter: maximum number of iterations
Output: $\mathcal{D}$: IDIs set
Const : m: size of population, $r1$: crossover rate, $r2$: mutate rate

```
1  D:=∅;
2  X:=initPopulation(s,m);
3  iter:=0;
4  while iter ≤ max_iter do
5      Calculate fitness values for the chromosomes in X;
6      for x ∈ X do
7          if discriminationCheck(x)==True then
8              D:=D ∪ x;
9      if Len(D) ≠ 0 then
10         return D
11     for i ∈ [0, m] do
12         if isBestFitness(X[i]) then
13             Continue;
14         Select x1, x2 using tournament selection;
15         X[i] := Crossover(x1, x2, r1);
16         X[i] := Mutate(X[i], r2);
17     iter += 1
18 return D
```

1) Overview: Figure 2 provides an overview of GRFT. To ensure diverse instances, we first cluster the original training dataset using algorithms like K-Means [29] and select seed samples from the clusters. GRFT then discovers IDIs from each seed through two phases: global generation and local generation. The global phase takes the seed samples as input and employs a genetic algorithm (GA) to efficiently search for more IDIs through population initialization, selection, crossover, and mutation operations. Based on the globally generated IDIs, the local stage aims to discover IDIs near them through local perturbation and search iterations.

2) Global Generation: Existing studies [13], [15], [23] often generate IDIs by guiding the search process using the gradient of the loss function or model output with respect to input x, which can be time-consuming. Moreover, findings in Section II-C indicate that improving the global phase leads to greater performance gains. Motivated by the high convergence speed and strong search capabilities of GA, we adopt GA in the global phase. Algorithm 1 presents the GA procedure for generating IDIs, with its key components detailed as follows.

Population Construction This step constructs the initial

Algorithm 2: Tournament Selection

Input : $fitness$: fitness values, X: current population, $tournament\_size$: size of tournament
Output: $x1, x2$: selected samples for crossover

```
tournament1 := randomSelect(population, tournament_size);
tournament2 := randomSelect(population, tournament_size);
x1 := getBestFitnessIndvidual(fitness, tournament1);
x2 := getBestFitnessIndvidual(fitness, tournament2);
return x1, x2
```

population based on a seed sample. In this paper, we focus on tabular datasets. For the encoding of GA, we consider a whole sample instance as a chromosome, and each attribute as a gene. Given a sample x, we generate a new sample x' through random perturbation of non-protected attributes chosen from x. Since the attributes are all preprocessed as categorical values in tabular samples, the perturbation is done by increasing or decreasing the value by s_g unit (random select from $[-c, c]$). c is used to constrain the difference between the attribute value of x and x'. The generated new chromosome will be checked to ensure that each attribute value complies with the constraints of the dataset. Attribute values that exceed the valid range will be clipped. As the initial step (Line 2 in Algorithm 1), we randomly generate m samples as the initial population.

Fitness Function Intuitively, if two instances differ only in protected attributes, a larger discrepancy in their model outputs suggests that these attributes significantly influence the model's inference. This implies that such instances are more prone to becoming discriminatory under slight perturbations. Based on this intuition, we design a fitness function $fitness(x)$ for a sample x as shown in Equation (1). It quantifies the absolute difference between the model's output probabilities for x and x_k using L_1-norm. x_k is derived from x and n is the number of samples that differ from x only in the protected attributes. The higher the fitness value, the more likely the sample x is to be mutated into IDIs. In each search iteration, this function is used to calculate the fitness value for each chromosome in the population X (Line 5 in Algorithm 1).

$$fitness(x) = \sum_{k=0}^{n} ||f(x) - f(x_k)||_1. \tag{1}$$

Selection This step selects high-quality chromosomes to generate the next population based on fitness score. In this paper, we adopt the tournament strategy, where the chromosome with the best fitness score in each tournament is selected for crossover (Line 14 in Algorithm 1). Algorithm 2 details the selection process. Specifically, $tournament\_size$ chromosomes are randomly selected from the top 10% ranked individuals in the current population as a tournament. Then, the chromosomes with the highest fitness score are selected from $tournament1$ and $tournament2$ respectively.

Crossover, and Mutation The crossover randomly exchanges the attribute values under the correspondence index (Line 15 in Algorithm 1). Next, the mutator randomly selects

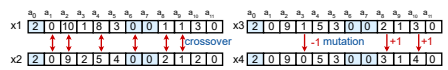

Fig. 3: The example of crossover and mutation

three non-protected attributes to increase or decrease s_g units (same as population construction) at a predefined mutation rate $r2$ (Line 16 in Algorithm 1). The mutated chromosome is then checked to ensure compliance with dataset constraints, and any attribute values exceeding the valid range are clipped.

Figure 3 illustrates an example of crossover and mutation. Chromosomes x_1 and x_2, used for crossover, are generated based on the dataset Census Income. The protected attributes a_0, a_6, and a_7 correspond to age, race, and gender, respectively, and are shown in the blue boxes. By randomly crossing the attributes $\{a_1, a_2, a_4, a_6, a_9\}$ of x_1 and x_2, a new chromosome x_3 is obtained. Then, the non-protected attributes a_3,a_8, and a_{10} of x_3 are mutated to 0, 3, and 4, respectively, resulting in the final new chromosome x_4.

3) Local Generation: Based on the IDIs found in the global phase, the local phase aims to explore the IDIs near them. Existing methods that rely on gradient calculations and iterative searching are time-consuming. However, our more effective global search algorithm allows us to adopt an efficient and less sophisticated search strategy in the local phase while maintaining high performance. Furthermore, neighbors of IDIs are often also likely to be discriminatory. Therefore, we employ a lightweight *random* search in the local phase.

Algorithm 3 shows the process of local generation. For each discriminatory instance d generated in the global phase, this phase iteratively searches through max_iter random samples (Line 3). In each iterative search, one non-protected attribute of d is randomly selected (Line 6) and either increased or decreased by 1 (Lines 7 and 10). The mutated instance is then checked to determine if it is still discriminatory (Line 11). If it is no longer discriminatory, the search continues from the original discriminatory instance d_0 (Line 14).

B. Repairing Individual Discrimination

The fitness score (Equation (1)) used in GA typically quantifies the degree of model unfairness. Higher fitness scores indicate a greater likelihood of discriminatory behavior, while lower scores suggest better fairness. To improve fairness, we reduce the model's tendency to produce high fitness scores by adding a loss term. We use L_{cls} to denote the cross entropy loss item and the retraining loss item could be revised as:

$$L = L_{cls} + \lambda \sum_{x}^{x \in X} fitness(x), \tag{2}$$

where λ is used to modulate the weight of the two items.

This revised loss function integrates traditional classification accuracy with a fairness-focused term, ensuring strong performance on the primary task while reducing biased behavior. Note that we only need to sample x from the original training data X, which eliminates the need for a large volume of generated IDIs.

Algorithm 3: Local Generation

Input : $\mathcal{D}_g$: IDIs generated in global phase, max_iter: maximum number of iterations, np: non-protected attributes
Output: $\mathcal{D}_l$: IDIs set

```
D_l:=∅;
iter:=0;
for d ∈ D_g do
    while iter ≤ max_iter do
        d_0:=copy(d);
        attribute:= randomSelect(np);
        if random.uniform() ≤ 0.5 then
            d[attribute] += 1;
        else
            d[attribute] -= 1;
        if discriminationCheck(d) == True then
            D_l:=D_l ∪ d;
        else
            d:=d_0;
        iter += 1
```

IV. Evaluation

In this section, we evaluate the performance of the proposed testing and repair method. We first outline the experimental setup and then, we introduce our evaluation aiming to answer the following research questions:

- **RQ1**: How does GRFT outperform other baselines across vanilla, repaired, and quantized models?
- **RQ2**: How does GRFT perform in both the global and local phases?
- **RQ3**: How efficient is GRFT?
- **RQ4**: How effective is the proposed repair method?

A. Experimental Setup

1) Datasets and Models.: We conduct experiments on five popular datasets including Census Income [30], Bank Marketing [30], German Credit [30], COMPAS [31], LSAC [32]. These tabular datasets are commonly used in individual fairness testing [15], [21], [23]. ❶ **Census Income.** The dataset, created by Barry Becker from the 1994 Census database, contains 48,842 instances and 14 attributes. The original aim is to determine whether a person makes over $50K a year. Among these 14 attributes, race, gender, and age are defined as protected attributes. ❷ **Bank Marketing.**The dataset contains over 45,000 instances and 16 attributes, with age being the only protected attribute. The original aim of this dataset is to assess whether a bank term deposit product would be subscribed to (yes) or not (no). ❸ **German Credit.** This dataset provides an assessment of creditworthiness based on personal and financial records. It contains 1,000 examples with 20 attributes. ❹ **COMPAS**. COMPAS (Correctional Offender Management Profiling for Alternative Sanctions) is a commercial algorithm used by judges and parole authorities to predict recidivism. A 2-year follow-up study showed that the algorithm is biased against black inmates and favors white defendants. ❺ **LSAC.** The LSAC dataset tracks students who entered law school in the fall of 1991 through three or more years of law examinations. National race- and gender-specific bar passage data are available for analysis and study. We train six-layer fully connected DNN models following prior work [15], [23], [26]. More details on our website [33].

2) Testing Baselines: We select 5 state-of-the-art testing approaches applicable for DNNs to compare with GRFT. The white-box baselines include ADF [23], EIDIG [15], Neuron-Fair [14], and the black-box baselines include ExpGA [14] and LIMI [13]. The white-box method DICE [25] is not selected because it uses the same gradient-guided strategy as ADF, with Shannon entropy to quantify individual discrimination (QID), generating multiple non-IDIs in the global phase (high QID value). This greatly increases time costs in the local phase, taking over four times longer for only a slight increase in IDIs. The results are available on our website [33].

3) Repairing Baselines: Following Faire [26], we select the following repairing baselines for comparison. ❶ **IDIs retraining** [15], [23]. ADF [23] and EIDIG [15] incorporate their generated IDIs into the training data to retrain the model and improve fairness. As shown in EIDIG, it outperforms ADF in retraining performance; thus, we primarily use EIDIG to generate retraining data. Following EIDIG, all IDIs across protected attribute combinations are added to the training data, forming the baseline M_{dis}. ❷ **Flipping-based retraining [26].** A basic method for augmenting the training data involves flipping the protected attributes of each instance (*e.g.*, generating a new instance by changing the gender from woman to man). During the learning process, the model learns to be insensitive to the protected attributes because the ground truth remains unchanged while only the protected attributes vary. We denote this baseline as M_{flip}. ❸ **Multitask Learning [34]**. A common approach to mitigate protected features uses a multi-task setting, training a protected attribute classifier alongside the original task. The loss function minimizes the original cross-entropy loss ($\mathcal{L}_1$) while maximizing the classifier's cross-entropy loss ($\mathcal{L}_2$). The final loss is $\mathcal{L} = \lambda_1\mathcal{L}_1 - \lambda_2\mathcal{L}2$. We denote this method as M_{mt}. ❹ **Faire [26].** This method identifies protected neurons and non-protected neurons. A condition layer is added to penalize protected neurons and promote non-protected neurons at each layer. The model is then fine-tuned using the original data, freezing hidden layers' weights and training only the condition layers, denoted as M_{Faire}. ❺ **CARE [35].** This method localizes faulty neurons through causality analysis and optimizes their parameters to mitigate misbehavior, denoted as M_{CARE}. ❻ **RULER [36].** This method balances accuracy and fairness by employing a two-phase training procedure with an iterative adversarial approach, denoted as M_{RULER}.

4) Evaluation Metrics: We design the following metrics for our evaluation: ❶ **Discriminatory Instance Number**. This metric tracks the number of IDIs identified, serving as both a testing and repair metric. For testing, a higher count indicates thorough detection of fairness issues. For repair, the goal is to reduce this number, reflecting improved model fairness. ❷ **Time**. This metric measures the total time needed for fairness

testing to identify IDIs. It covers both global and local phases. A shorter time indicates higher efficiency, which is important for practical applications. ❸ **Accuracy**. The accuracy of the repaired model is an important indicator for measuring its utility. The objective of the repair method is to improve the individual fairness of the given models without sacrificing too much accuracy.

5) Parameter Setting: Recall that GRFT involves 5 important parameters, including m and c for population construction, $tournament\_size$ for sample selection, $r1$ for the crossover, and $r2$ for the mutation. We configure m, c, $tournament\_size$, $r1$, $r2$ to 100, 5, 3, 0.5, 0.5 according to our experimental experience for the tabular dataset. The parameter λ is set to 1 for the repair method. Additionally, before the global generation phase, we employ the K-Means algorithm to cluster the training set. Across all experiments, following ADF and EIDIG, we set the number of clusters to 4 and select 1,000 samples as seed, and set the maximum number of iterations for the global phase and local phase to 10 and 1,000, respectively. Additionally, we use the best settings reported in the papers for all testing and repair baselines. We take an average of 10 multiple runs for all experiments to ensure robust results. Under most parameter settings, the p-values < 0.02. It shows our results are statistically significant.

B. RQ1: Performance of GRFT across Vanilla, Repaired, Quantized Models

1) Vanilla Models: The number of IDIs found by each fairness testing method on vanilla models are shown in Table I. We can see that all the testing methods yield different results across various datasets and attribute combinations. ADF and EIDIG detect the most IDIs on the Bank dataset, while ExpGA and GRFT detect the most on the LSAC dataset. However, GRFT consistently outperforms all other methods in detecting more IDIs across every dataset and attribute combination. Its superior performance is particularly notable in complex attribute combinations and larger datasets, such as the LSAC dataset and the attribute $g\&r$. Furthermore, our black-box testing method, GRFT, detects an average of 358,843 IDIs across all datasets and attribute combinations, significantly higher than the next best method, EIDIG, which identifies 145,420. These results demonstrate GRFT's robustness and effectiveness in identifying IDIs.

2) Repaired Models: Table II compares testing methods across repaired models, showing the average number of IDIs identified across all protected attributes. A "–" indicates no IDIs were found. More details are available on our website [33].

We observe that on most repaired models, gradient-based white-box methods like ADF, EIDIG, and NeuronFair outperform black-box methods such as ExpGA and LIMI. However, on M_{flip}, LIMI detects more IDIs than ADF and EIDIG. This may be because models closer to fairness have smoother gradients, making them less effective for guiding IDI searches. Notably, all methods failed to find IDIs on M_{mt} for the Credit and COMPAS datasets due to the lower model accuracy, which produced identical outputs for all inputs. LIMI could not find IDIs on M_{Faire} because the surrogate decision boundary could not be generated due to the instability of GAN models.

TABLE I: Number of IDIs found by each fairness testing method on vanilla models

Dataset	Attr	ADF	EIDIG	NeuronFair	ExpGA	LIMI	GRFT
Census	a	229,544	244,015	288,467	26,117	3,886	**496,374**
	g	50,331	57,130	22,366	2,831	30,194	**222,461**
	r	45,150	53,986	37,366	33	17,848	**400,136**
	a&g	313,364	339,186	313,932	28,402	19,715	**857,909**
	a&r	295,721	324,551	304,666	29,075	14,476	**285,600**
	r&g	125,078	145,463	83,996	8,978	33,334	**599,456**
	Avg	176,531	194,055	175,132	15,906	19,908	**476,989**
Bank	a	236,844	257,761	198,884	9,023	16,809	**138,389**
Credit	a	197,949	229,846	217,692	19,232	0	**497,263**
	g	30,052	31,285	136,119	0	10,724	**269,214**
	a&g	238,376	259,813	265,393	23,787	26,944	**336,819**
	Avg	155,459	173,648	206,401	21,509	18,834	**367,765**
COMPAS	g	170	244	118	272	11	**382**
	r	150	158	81	84	29	**261**
	g&r	257	369	153	374	41	**475**
	Avg	192	257	117	243	27	**372**
LSAC	g	77,172	30,371	20,825	14,389	774	**265,654**
	r	168,425	135,550	123,468	14,070	1,399	**614,612**
	g&r	246,745	217,002	195,091	30,744	1,563	**756,486**
	Avg	164,114	127,641	113,128	19,734	1,245	**545,584**
Total		140,958	145,420	138,038	13,827	11,849	**358,843**

Compared to existing methods, GRFT identifies the most IDIs across all repaired models, with a significant margin over the next best method. For example, in M_{dis}, GRFT finds 405,760 IDIs for the LSAC dataset, a considerable improvement over the best gradient-based method, ADF, which finds 86,798 IDIs. Similarly, in M_{flip}, GRFT identifies 7,217.4 IDIs, significantly surpassing other methods. These results underscore GRFT's robustness in handling diverse fairness-enhanced models and highlight its potential as a powerful tool for fairness testing in machine learning.

3) Quantized Models: To assess the impact of quantization on model fairness, we apply quantization and testing to the repaired models. For each model, we consider three quantization levels (*i.e.*, 1%, 50%, and 100%). We then use all fairness testing methods to detect IDIs. The results show that GRFT outperforms other baselines across all datasets and quantized models. Table III shows the number of IDIs found by GRFT on the best-performing quantized models (*i.e.*, quantized M_{flip}), where 0% indicates the model that is not quantized. More detailed results are shown in our website [33].

From the table, it is evident that quantization impacts the number of IDIs detected. Specifically, as the quantization level increases, the number of IDIs generally increases across most datasets and attributes. For instance, in the Census dataset, the attribute combination of age and race (a&g) shows a steady increase in IDIs from 6,236 at 0% quantization to 6,704 at 100%. Similarly, in the Credit dataset, the attribute combination of gender and age (a&g) shows an increase from 9,701 at 0% quantization to 29,808 at 100%.

This pattern suggests that quantization generally exacerbates the fairness issues in the models, leading to an increase in

TABLE II: Number of IDIs found by fairness testing methods on repaired models

Dataset	M_{dis}						M_{mt}					
	ADF	EIDIG	NeuronFair	ExpGA	LIMI	**GRFT**	ADF	EIDIG	NeuronFair	ExpGA	LIMI	**GRFT**
Census	23,296	18,588	13,539	2,058	2,884	**92,804**	342,338	367,000	264,159	19,378	18,571	**372,560**
Bank	99,100	86,491	47,090	10,578	14,845	**91,530**	330,882	323,730	107,467	109	-	**233,889**
Credit	21,111	14,058	19,202	3,059	5,392	**144,650**	-	-	-	-	-	**-**
COMPAS	499	744	280	848	16	**928**	-	-	-	-	-	**-**
LSAC	86,798	56,013	39,808	16,284	654	**405,760**	273,741	346,400	145,676	24,801	550	**1,120,487**
Avg	46,160.8	35,178.8	23,983.8	6,565.4	4,758.2	**147,134.4**	315,653.7	345,710	172,434	14,762.7	9,560.5	**575,645.4**

Dataset	M_{flip}						M_{Faire}					
	ADF	EIDIG	NeuronFair	ExpGA	LIMI	**GRFT**	ADF	EIDIG	NeuronFair	ExpGA	**LIMI**	**GRFT**
Census	534	442	681	326	884	**7,284**	82,099	86,543	107,895	3,777	12,098	**309,056**
Bank	829	558	602	418	1,504	**7,931**	78,825	78,777	174,818	42	1,027	**145,994**
Credit	3,237	2,980	8,307	1,382	4,527	**14,773**	11,191	11,204	28,349	-	-	**26,902**
COMPAS	9	10	7	155	25	**241**	194	199	148	4,021	10	**468**
LSAC	287	211	457	821	184	**5,858**	280,400	275,863	99,930	10,924	-	**909,300**
Avg	979.2	840.2	2010.8	620.4	1,424.8	**7,217.4**	90,541.8	90,517.2	82,228	3,752.8	4,378.4	**278,344.1**

Dataset	M_{CARE}						M_{RULER}					
	ADF	EIDIG	NeuronFair	ExpGA	LIMI	**GRFT**	ADF	EIDIG	NeuronFair	ExpGA	LIMI	**GRFT**
Census	40,060	51,266	46,210	3,990	6,063	**148,562**	79,760	112,405	89,948	12,284	6,535	**219,222**
Bank	131,750	149,358	172,961	5,208	4,601	**98,223**	22,596	30,977	57,660	2,833	45	**94,083**
Credit	162,098	137,475	151,922	17,066	16,604	**282,252**	62,200	68,417	133,231	4,757	32,719	**204,985**
COMPAS	108	144	57	7	4	**151**	173	253	86	155	6	**247**
LSAC	80,350	73,052	58,534	11,482	816	**454,732**	102,921	93,076	61,716	8,415	795	**463,682**
Avg	82,873.2	82,259	85,936.8	7,550.6	5,617.6	**196,784**	53,530	61,025.6	68,528.2	5,688.8	8,020	**196,443.8**

TABLE III: IDIs found by GRFT on quantized M_{flip}

Dataset	Attr	0%	1%	50%	100%
Census	a	5,808	6,037	5,957	5,685
	g	4,618	4,855	4,729	4,288
	r	4,241	6,096	4,857	6,006
	a&g	6,236	8,017	7,967	6,704
	a&r	13,012	13,571	12,937	13,629
	r&g	9,790	9,535	8,972	9,751
Bank	a	7,931	8,955	9,777	8,884
Credit	a	16,572	65,763	70,433	62,142
	g	18,048	56,599	54,512	58,467
	a&g	9,701	31,559	35,831	29,808
COMPAS	g	67	63	61	63
	r	8	50	52	52
	g&r	49	66	65	65
LSAC	g	529	4,133	3,808	4,216
	r	7,319	5,251	5,721	5,298
	g&r	9,727	10,740	8,976	8,648

discriminatory instances detected. The effect is particularly pronounced in datasets like Credit, where the number of IDIs increases significantly with higher quantization levels.

Answer to RQ1: GRFT consistently outperforms all testing baselines in detecting more IDIs across vanilla, repaired, and quantized models. Moreover, quantization seems to harm model fairness, as indicated by the rise in IDIs with higher quantization levels.

C. RQ2: Performance on Global Phase and Local Phase

To understand the reasons for GRFT's effectiveness, we conducted a detailed analysis of its performance in the global and local phases.

1) Global Phase: Table IV presents the number of IDIs detected by each testing method during the global phase across vanilla, repaired, and quantized models. Due to space limitations and the similar results observed on the quantized and repaired models, we only present results for the top 4 testing baselines and the best-performing quantized M_{flip}. LIMI is excluded as it only contains one random exploration phase. A "–" in the table indicates that no IDIs are found.

We can see that, in the global phase, GRFT significantly outperforms other state-of-the-art methods across vanilla, repaired, and quantized models. Specifically, for 1,000 initial seeds and the M_{flip}, GA-based GRFT detects an average of 2,652 IDIs per dataset, significantly surpassing ADF (*i.e.*, 63), EIDIG (*i.e.*, 59), NeuronFair (*i.e.*, 112), and ExpGA (*i.e.*, 84). GRFT can detect more than 1,000 IDIs, which exceeds the number of seeds. This is because, in each iteration, the population size is set to 100. Therefore, for each seed, more than one discriminatory instance can be detected. In contrast, existing gradient-based methods generate at most one discriminatory instance per seed. For the Census, Credit, COMPAS, and LSAC datasets, ExpGA produces the fewest IDIs as it does not search for IDIs but instead uses an interpretability model to generate high-quality samples. These results demonstrate that GRFT significantly outperforms other testing methods in the global phase, primarily due to the superior search capabilities and fast convergence of the GA.

2) Local Phase: Despite using only a random search strategy without directional guidance, GRFT detects a significant number of IDIs in the local phase. For vanilla models, GRFT

TABLE IV: Number of IDIs found by each fairness testing method in the global phase

Dataset	M_{van}				M_{dis}				M_{mt}				M_{flip}			
	ADF	EIDIG	NeuronFair	**GRFT**	ADF	EIDIG	NeuronFair	**GRFT**	ADF	EIDIG	NeuronFair	**GRFT**	ADF	EIDIG	NeuronFair	**GRFT**
Census	429	526	521	**5,072**	301	225	221	**2,108**	630	667	485	**3,431**	71	66	109	**1,439**
Bank	236	576	562	**4,229**	653	656	328	**3,276**	835	845	278	**4,747**	100	89	81	**7,931**
Credit	439	422	551	**5,107**	148	91	264	**3,109**	-	-	-	**-**	63	78	267	**2,438**
COMPAS	109	82	50	**269**	68	114	72	**442**	-	-	-	**-**	4	5	4	**55**
LSAC	592	562	520	**4,565**	557	403	365	**3,454**	624	775	399	**4,976**	78	59	98	**1,398**
Avg	361	434	441	**3,848**	345	298	250	**2,478**	418	457	232	**2,631**	63	59	112	**2,652**

Dataset	M_{Faire}				M_{CARE}				M_{RULER}				Quantized M_{flip}			
	ADF	EIDIG	NeuronFair	**GRFT**	ADF	EIDIG	NeuronFair	**GRFT**	ADF	EIDIG	NeuronFair	**GRFT**	ADF	EIDIG	NeuronFair	**GRFT**
Census	315	322	334	**4,749**	228	291	329	**3173**	359	523	415	**3492**	71	66	110	**1,381**
Bank	288	288	516	**4,309**	608	718	449	**3939**	158	249	325	**3145**	111	93	93	**1,501**
Credit	30	30	111	**204**	439	404	500	**4260**	212	261	447	**3701**	64	78	262	**2,469**
COMPAS	336	219	44	**22**	34	55	24	**133**	43	68	32	**188**	4	5	3	**53**
LSAC	641	2,388	238	**5,133**	459	484	399	**3976**	574	579	461	**3970**	77	57	104	**1,342**
Avg	322	649	249	**2,883**	354	390	340	**3,096**	269	336	336	**2,899**	65	60	114	**1,349**

detects 301,972 (*i.e.*, 305,820-2,652) instances on average, accounting for 98.74% of the total instances detected, while for M_{flip}, it detects 4,625 (*i.e.*, 7,217-2,652) instances, comprising 63.63%. In comparison, ADF, EIDIG, and NeuronFair detect 916, 781, and 1,989 instances in the local phase, contributing 93.56%, 92.97%, and 94.42% of their respective totals. Notably, ExpGA generates fewer IDIs, as its global phase provides too few high-quality seeds, highlighting the effectiveness of GRFT in leveraging even a simple random search for local phase detection.

3) Ablation Analysis : To evaluate the contributions of the global and local phases, we replace the search algorithms in GRFT with those from the two best-performing baselines, EIDIG and NeuronFair, forming variants shown in the first row of Table V. Specifically, GRFT&NeuronFair represents the variant combining GRFT's global search with NeuronFair's local search, while EIDIG&GRFT represents the variant combining EIDIG's global search with GRFT's local search.

For the evaluation, 1,000 samples from the Census dataset were randomly selected as seeds, with 10 and 100 iterations set for the global and local phases, respectively. Each experiment was repeated three times, and the average results are reported in Table V, where #IDIs denotes the number of IDIs found, and #IDIs_Per_Second measures efficiency, representing the average number of IDIs generated per second. The metric #IDIs_Per_Second was introduced to ensure a fair comparison, as different methods require varying amounts of time to complete the same number of iterations.

When replacing Random in GRFT with the local search algorithm in NeuronFair and EIDIG, more IDIs were generated after 100 iterations (13,079 vs. 19,738 vs. 23,039). However, the time required increased significantly (1.79s vs. 2,496.92s vs. 705.82s), making them 1,394 and 393 times slower. Conversely, when replacing NeuronFair and EIDIG with simple Random, they became much faster (22.43s and 28.58s) but produced fewer IDIs. These results highlight that enhancing the global phase while using lightweight search in the local phase leads to higher IDIs, especially when the SAME time budget is applied.

TABLE V: Performance of variant methods

	GA&Random (GRFT)	GRFT& NeuronFair	GRFT& EIDIG	EIDIG& GRFT	NeuronFair& GRFT
#IDIs	13,079	19,738	23,039	2,340	1,944
Time (Seconds)	1.79	2,496.92	705.82	22.43	28.58
#IDIS_Per_Second	7316.75	7.90	32.64	104.32	68.02

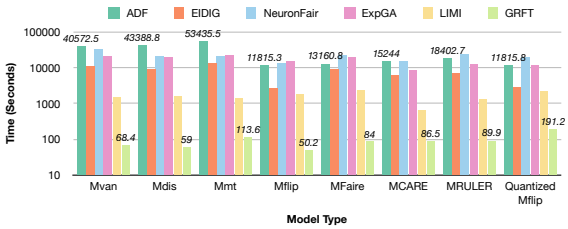

Fig. 4: Comparison of total time required by fairness testing methods across vanilla, repaired, and quantized models

Answer to RQ2: This significant increase in generated global IDIs allows us to adopt an efficient and less sophisticated random search strategy in the local phase while maintaining high effectiveness.

D. RQ3: The Efficiency of GRFT

To evaluate the efficiency of GRFT, we calculated the total time required for each testing method to complete the search for each model and 1,000 initial seeds. Figure 4 shows the average result on vanilla, repaired, and quantized models. Due to space limitations and the similar results observed on the quantized and repaired models, we only present results for the best-performing quantized M_{flip}. More detailed results can be found on our website [33]. We can see that for all models, GRFT completes the search in under 200 seconds, while ADF, EIDIG, and NeuronFair take over 10,000 seconds (at least 50 times longer than GRFT). Even the relatively faster LIMI requires at least 1,000 seconds. It is worth noting that the times required by baselines are higher than reported in their papers because we measure the total time to generate all IDIs from 1,000 seeds (10 global and 1,000 local iterations), whereas the papers report the time for generating only 1,000 IDIs. Compared to the vanilla models, the testing methods

generally complete the search faster on repaired models, as fewer IDIs are detected in the global phase in fairer models, limiting the number of searches in the local phase. These results indicate that GRFT completes the search process faster than all baselines, at least 5 times faster than GAN-based LIMI and even 50 times faster than gradient-guided methods.

Additionally, Table I shows that GRFT identifies more IDIs across most datasets and models compared to all baselines. These results indicate that GRFT can discover more IDIs in less time. For instance, on vanilla models, GRFT identifies 358,843 IDIs in an average of 68.4 seconds, while the next best method, EIDIG, takes 11,457.6 seconds to find 145,420 instances. Similarly, GRFT takes only 84 seconds to discover 3.07 times more IDIs (*i.e.*, 278,344) than EIDIG does in the models repaired by Faire.

We believe the primary reason for GRFT's high performance is the combination of the GA and the random iterative search algorithm. First, the GA significantly enhances the effectiveness of the global phase, allowing us to adopt an efficient and less complex search strategy in the local phase while maintaining high effectiveness. Second, the use of an unguided, random iterative search in the local phase reduces time costs by avoiding the complex and time-consuming gradient computations required by existing methods and by processing all seeds in batches. In contrast, other baselines require mutations based on the gradient of input pairs, making batch processing a challenge.

Answer to RQ3: GRFT significantly reduces the time required to identify IDIs because they do not require gradient calculations, demonstrating superior efficiency over other methods.

E. RQ4: Performance of Our Repair Method

As shown in Table II, a considerable number of IDIs can still be detected in M_{dis}, M_{mt}, and M_{faire}. Particularly, more IDIs can be detected in M_{mt} than in the vanilla models. For example, ADF detects about 315,654 IDIs in M_{mt}, which is 2.24 times the number found in the vanilla models (*i.e.*, 140,958). This may stem from the instability of the multitask learning paradigm in the repair method. While it improves fairness on the original test set, it may fail to enhance robustness and can even reduce it if misdirected. As observed in Faire, the adversarial branch often converges to fixed outputs during training, failing to counter discriminatory patterns effectively. Notably, for M_{mt}, no IDIs are detected on the Credit and COMPAS datasets due to the lower accuracy resulting in the same output for all inputs.

As a comparison, the number of detected IDIs in models repaired by our method is shown in Table VI. We can see that our repair method is highly effective in reducing the number of IDIs across all datasets. Particularly in the Census, Bank, COMPAS, and LSAC datasets, ADF, EIDIG, and NeuronFair detected fewer than 20 IDIs. ExpGA was unable to find discriminatory inputs in the Bank and Credit datasets. While GRFT identifies a relatively higher number of IDIs in our repaired models due to its rigorous nature, the overall reduction in bias achieved by our repair method surpasses that of existing repair methods. For example, GRFT on average discovers 2,850.6 IDIs in the models repaired by our method, which is a 60.50% reduction compared to the flipping-based retrained models. These findings clearly demonstrate that our repair method is effective in mitigating bias and improving the fairness of deep learning models, making it a valuable tool in the development of equitable AI systems.

TABLE VI: Total discriminatory instance number found in our repair models

Dataset	ADF	EIDIG	NeuronFair	ExpGA	LIMI	GRFT
Census	7	14	17	602	56	1,145
Bank	12	7	6	0	45	400
Credit	1,898	2,554	2,062	0	3,180	12,471
COMPAS	4	4	9	18	0	31
LSAC	8	4	12	7	25	206
Avg	**385.8**	**516.6**	**421.2**	**209**	**826.5**	**2,850.6**

TABLE VII: Accuracy of vanilla and repaired models

Dataset	M_{van}	M_{dis}	M_{mt}	M_{flip}	M_{Faire}	M_{CARE}	M_{RULER}	**Ours**
Census	0.842	0.844	0.814	0.844	0.828	0.813	0.830	**0.841**
Bank	0.893	0.891	0.88	0.891	0.884	0.888	0.894	**0.892**
Credit	0.762	0.745	0.673	0.712	0.705	0.764	0.737	**0.744**
COMPAS	0.655	0.658	0.530	0.650	0.628	0.649	0.638	**0.646**
LSAC	0.868	0.857	0.841	0.858	0.828	0.862	0.850	**0.867**
Avg	0.804	0.799	0.748	0.791	0.775	0.795	0.790	**0.798**

In addition, the accuracy of vanilla and repaired models are shown in Table VII. From the table, we can observe that compared to the vanilla model, most repaired models showed a decrease in accuracy. Specifically, the M_{dis}, M_{CARE}, and our repaired models maintain relatively high accuracy, with an average accuracy exceeding 79.4% across all datasets, which represents a decrease of less than 1% compared to the vanilla models. However, M_{mt} achieves an average accuracy of 74.8% across all datasets, representing a decrease of approximately 6% compared to the vanilla models. Specifically, on the COMPAS dataset, the accuracy of M_{mt} drops to 0.53, a 12% reduction compared to the vanilla models, significantly impairing the model's performance.

These results indicate that our repair method preserves the model's original accuracy while resulting in fewer new IDIs. This demonstrates that our repair method effectively addresses fairness issues in models.

Answer to RQ4: Our repair method not only preserves the original accuracy of the model but also results in fewer new IDIs.

V. Discussion

A. Quality of Synthetic Data

The perturbations of attributes may produce unrealistic inputs, leading to false positives and an overestimation of discrimination. To address this limitation, following previous work [15], [23], each input attribute is constrained to its minimum and maximum range in the training dataset.

Furthermore, we evaluate the perturbation distance (measured with cosine distance) and distribution distance (measured with Maximum Mean Discrepancy) between the generated IDIs and the original data by randomly selecting 1,000 seeds from the Census dataset and generating IDIs with all methods over 10 rounds, repeating each experiment 3 times. The results show that the average perturbation distance across all methods ranged from 0.038 to 0.058, with GRFT achieving a value of 0.046, which is closer to the lower bound of the range, demonstrating comparable quality to the baselines. Similarly, the average distribution distance, ranging from 0.007 to 0.014, indicates that all methods preserve the original data distribution. This is due to the shared mutation strategy that introduces small, valid perturbations. Notably, we also analyzed how many IDIs found by GRFT are new or previously discovered by the baselines, as shown in Figure 5. GRFT uniquely discovered 72,729 IDIs, significantly more than the baselines, while only 259 instances are commonly identified by ADF, EIDIG, or NeuronFair. This advantage arises from GRFT's more efficient global search strategies, which allow it to explore a broader input space.

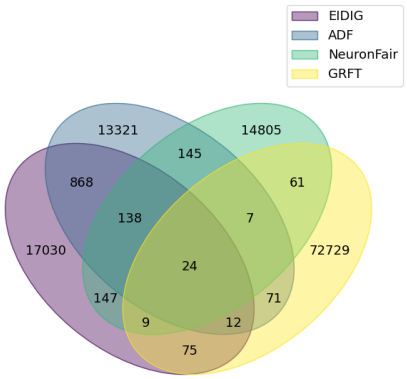

Fig. 5: IDIs generated by EIDIG, ADF, NeuronFair, and GRFT

B. Deeper Insights into Performance

Our method's high performance is due to two main factors. First, the GA in the global phase generates multiple discriminatory instances from each seed, boosting search effectiveness. Second, the random iterative search in the local phase avoids the time-consuming gradient calculations used by other methods. This combination allows our method to outperform existing approaches in both speed and effectiveness. Notably, combining the GA with a gradient-based local search might uncover more discriminatory instances, but it would significantly increase the time cost.

C. Generalizability

GRFT is a black-box testing method that can be easily extended to other models (*i.e.*, CNN) and data types (*e.g.*, text and image datasets). Unlike tabular data, the attributes of the image and text are difficult to modify directly from the input domain. Therefore, we slightly adjust the modification of the protected attribute in GRFT following NeuronFair and ExpGA, respectively.

TABLE VIII: The testing performance of each method on the CelebA and SST datasets.

Dataset	CelebA					SST	
Method	ADF	EIDIG	NeuronFair	LIMI	**GRFT**	ExpGA	**GRFT**
#IDIs	16	16	87	75	**543**	8	**298**
Time(Seconds)	3,501	3,526	2,640	647	**2,879**	1,764	**1,545**
Time per IDI	218.81	220.37	30.34	8.62	**5.30**	220.5	**5.18**

For a given image x and classifier $f(x)$, there are three steps following NeuronFair: First, we build a sensitive attribute classifier $f_{sa}(x)$ that can distinguish the image's protected attributes (*e.g.*, gender). Next, we use the classic FGSM adversarial attack [37] to modify the images' protected attributes and flip their predicted results by generating Δ_{senatt} as follows:

$$\Delta_{senatt} = \epsilon \times sign(\frac{\partial f_{sa}(x)_{pred}}{\partial x}) \tag{3}$$

satisfying that $f_{sa}(x) \neq f_{sa}(x + \Delta_{senatt})$ where ϵ is a hyper-parameter to determine perturbation size, sign(·) is a signum function return -1, 0, or 1. Finally, we leverage GRFT to generate Δ_{bias} and then determine whether the instance pair $< x + \Delta_{bias}, x + \Delta_{senatt} + \Delta_{bias} >$ satisfy definition 2.

For a given text sample x containing a sensitive word $a_i \triangleright p$, following ExpGA, we modify the protected attribute by substituting a_i with a pair of semantically opposite words, $\tilde{a}_i$ and $\neg\tilde{a}_i$. For non-protected attributes, we randomly replace non-sensitive words with semantically similar words identified using the word embedding tool Glove [38].

In this study, we evaluate a ResNet-50 model trained on the image dataset CelebA [39] for smile detection and a 3-layer CNN model trained on the text dataset SST [40], with gender as the protected attribute. For the experimental setup, 1,000 seeds are randomly selected, and iterates are set to 10 and 100 in global and local phases, respectively. For the image dataset, only the global phase is applied, as IDIs generated in the local phase differ by only a few pixels from the global phase, contributing minimally to improving classifier fairness. During image mutation, $n \in [10, 50]$ pixels are randomly adjusted by $\pm 1/255$. Baselines are evaluated using the optimal settings from NeuronFair and ExpGA.

The results in Table VIII show that GRFT outperforms all baselines by identifying more IDIs with higher efficiency. On the image dataset, GRFT discovers an IDI 41.3x, 41.6x, 5.7x, and 1.63x faster than ADF, EIDIG, NeuronFair, and LIMI, respectively. LIMI is faster due to its pre-trained GAN but may be slow in real-world use. On the text dataset, GRFT generates 298 IDIs in 1,545 seconds, compared to only 8 IDIs in 1,764 seconds by ExpGA. Notably, other baselines are not included in this comparison as they do not support text datasets.

Furthermore, we measure the bias perturbation Δ_{bias} using the L_2-norm for generated image data. GRFT achieves a significantly lower Δ_{bias} (8.25) compared to the baselines, where ADF, EIDIG, LIMI, and NeuronFair have Δ_{bias} values of 291.45, 291.80, 140.37, and 282.87, respectively. This is because GRFT perturbs a smaller subset of pixels in each search step, while baselines modify larger pixel regions in

a single step. These results underscore GRFT's practical advantages in handling high-dimensional image and text data, with superior efficiency and effectiveness.

D. Future Work

While GRFT excels in fairness testing, identifying the root causes of DNN unfairness and enhancing IDI diversity remain open challenges. Clustering-based analysis reflects IDI diversity but does not directly correlate with the causes of DNN unfairness. Furthermore, the reliance on similar mutation techniques across methods limits input diversity, highlighting the need for future research on explanation methods and advanced mutation strategies.

E. Threats to Validity

Limited model and datasets. In this paper, we follow commonly used settings to primarily evaluate the effectiveness of GRFT on tabular data. However, as demonstrated in Section V-C, GRFT can be extended to other domains, showcasing its potential and generalizability to image and text data.

VI. Related Work

There is a substantial body of work focusing on testing and improving group fairness, where examples are grouped according to a particular sensitive attribute and statistics are calculated across groups. Notable studies include [16], [41]–[52]. However, evaluations of group fairness and individual fairness are fundamentally different [47]. In our paper, we focus on individual fairness.

A. Fairness Testing

Most existing individual fairness testing work uses a two-step approach that first perturbs random sample instances from the input dataset to find a discriminatory instance and then locally perturbs those instances to further generate biased test cases. These methods can be divided into two categories: black-box [13], [14], [20]–[22] and white-box [15], [23]–[25].

In the black-box setting, [20] define fairness and discrimination and develop THEMIS, a tool to generate efficient test suites for measuring discrimination. To address THEMIS's inefficiencies caused by the random sampling process, [21] propose AEQUITAS, which discovers IDIs through random exploration in the global phase and perturbs instances using three strategies in the local phase to generate more IDIs. [22] employs LIME [53] to generate local perturbation samples to build a decision tree and then analyzes each tree path to generate test inputs. Similar to [22], ExpGA [14] employs interpretable methods to collect high-quality initial seeds in the global phase and then adopt GA to search IDIs in the local phase. Recently, [13] proposes LIMI to generate more natural IDIs with the help of a generative adversarial network (GAN). However, training GANs is inefficient, and their performance is limited by the assumption that they accurately capture the decision boundary, which is not always true.

In the white-box setting, ADF [23] utilizes the gradient of the loss function to search IDIs near the decision boundary of DNN. EIDIG [15] enhances ADF by incorporating momentum into the global phase and improving the local phase with the effective vicinity explorer to boost exploration effectiveness. NeuronFair [24] and DICE [25] enhance performance by calculating gradients only for identified biased neurons and quantifying fairness based on protected information used in decision-making, respectively.

Though these methods have made progress in fairness testing, their extensive gradient calculations, iterative searches, and additional model training raise efficiency concerns. Moreover, they often improve performance by enhancing the local phase, focusing on more iterations per global IDI. In contrast, our GRFT focuses on enhancing the global phase with the strong search capabilities of GA to quickly identify many global IDIs and employs a simple random perturbation strategy in the local phase to reduce time costs.

B. Fairness Repair

A series of research [13], [15], [23] integrate generated IDIs to retrain models and improve fairness. However, they are inefficient due to the significant time overhead caused by the generation process. To avoid using generated discriminatory data, [35] propose CARE, a causality-based technique for repairing neural networks by localizing faults and using PSO to adjust the weights of identified neurons. Similarly, [26] adds a condition layer after each hidden layer to penalize neurons linked to protected features and promote those tied to non-protected features. This method modifies the model architecture but results in a greater accuracy trade-off. Unlike them, we improve model fairness by introducing a novel loss function term while ensuring accuracy on the original task.

VII. Conclusion

This paper investigates individual discrimination in DNNs and introduces GRFT as an effective testing method, along with a simple yet effective repair approach for mitigating discrimination. Extensive experiments involving six testing methods, 261 models, five datasets, and 16 sensitive attribute combinations demonstrate the generalizability of our approach. Further refinement and expansion of these methods can contribute to the development of fairer deep learning models.

Acknowledgment

This work was supported by the National Key Research and Development Program of China (2023YFB3107100), the National Research Foundation, Singapore, the Cyber Security Agency under its National Cybersecurity R&D Programme (NCRP25-P04-TAICeN), the National Research Foundation Singapore and DSO National Laboratories under the AI Singapore Programme (AISG Award No: AISG2-GC-2023-008), and the National Research Foundation, Prime Minister's Office, Singapore under the Campus for Research Excellence and Technological Enterprise (CREATE) programme. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore and Cyber Security Agency of Singapore.

REFERENCES

[1] Datamonsters, "10 applications of artificial neural networks in natural language processing," 2017. [Online]. Available: https://medium.com/@datamonsters/artificial-neural-networks-in-natural-language-processing-bcf62aa9151a

[2] TechCrunch, "Nearly 70% of us smart speaker owners use amazon echo devices," 2020. [Online]. Available: https://techcrunch.com/2020/02/10/nearly-70-of-u-s-smart-speaker-owners-use-amazon-echo-devices/

[3] X. Li, Z. Chen, J. M. Zhang, F. Sarro, Y. Zhang, and X. Liu, "Bias behind the wheel: Fairness testing of autonomous driving systems," *ACM Transactions on Software Engineering and Methodology*, 2024.

[4] Z. Sun, Z. Chen, J. Zhang, and D. Hao, "Fairness testing of machine translation systems," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 6, p. 156, 2024.

[5] A. Chouldechova, "Fair prediction with disparate impact: A study of bias in recidivism prediction instruments," *Big data*, vol. 5, no. 2, pp. 153–163, 2017.

[6] BBC News. (2021) Ai at work: Staff 'hired and fired by algorithm. [Online]. Available: https://www.bbc.com/news/technology-56515827

[7] BBCNews. (2020) Ibm abandons "biased" facial recognition tech. [Online]. Available: https://www.bbc.co.uk/news/technology-52978191

[8] B. Johnson and T. Menzies, "Unfairness is everywhere, so what to do? an interview with jeanna matthews," *IEEE Softw.*, vol. 40, no. 6, pp. 135–138, 2023. [Online]. Available: https://doi.org/10.1109/MS.2023.3305722

[9] S. Biswas and H. Rajan, "Do the machine learning models on a crowd sourced platform exhibit bias? an empirical study on model fairness," in *ESEC/FSE '20: 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual Event, USA, November 8-13, 2020*, P. Devanbu, M. B. Cohen, and T. Zimmermann, Eds. ACM, 2020, pp. 642–653. [Online]. Available: https://doi.org/10.1145/3368089.3409704

[10] K. M. Habibullah, G. Gay, and J. Horkoff, "Non-functional requirements for machine learning: understanding current use and challenges among practitioners," *Requir. Eng.*, vol. 28, no. 2, pp. 283–316, 2023.

[11] K. M. Habibullah and J. Horkoff, "Non-functional requirements for machine learning: Understanding current use and challenges in industry," in *29th IEEE International Requirements Engineering Conference, RE 2021*, 2021, pp. 13–23.

[12] B. Kim, J. Wang, and C. Wang, "Fairquant: Certifying and quantifying fairness of deep neural networks," in *Proceedings of the 47th IEEE/ACM International Conference on Software Engineering, ICSE 2025*, 2025.

[13] Y. Xiao, A. Liu, T. Li, and X. Liu, "Latent imitator: Generating natural individual discriminatory instances for black-box fairness testing," in *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2023, Seattle, WA, USA, July 17-21, 2023*, R. Just and G. Fraser, Eds. ACM, 2023, pp. 829–841. [Online]. Available: https://doi.org/10.1145/3597926.3598099

[14] M. Fan, W. Wei, W. Jin, Z. Yang, and T. Liu, "Explanation-guided fairness testing through genetic algorithm," in *44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022*. ACM, 2022, pp. 871–882. [Online]. Available: https://doi.org/10.1145/3510003.3510137

[15] L. Zhang, Y. Zhang, and M. Zhang, "Efficient white-box fairness testing through gradient search," in *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ser. ISSTA 2021. New York, NY, USA: Association for Computing Machinery, 2021, p. 103–114. [Online]. Available: https://doi.org/10.1145/3460319.3464820

[16] J. Chakraborty, S. Majumder, and T. Menzies, "Bias in machine learning software: why? how? what to do?" in *ESEC/FSE '21: 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Athens, Greece, August 23-28, 2021*, D. Spinellis, G. Gousios, M. Chechik, and M. D. Penta, Eds. ACM, 2021, pp. 429–440. [Online]. Available: https://doi.org/10.1145/3468264.3468537

[17] S. Biswas and H. Rajan, "Fair preprocessing: towards understanding compositional fairness of data transformers in machine learning pipeline," in *ESEC/FSE '21: 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2021, pp. 981–993.

[18] Z. Chen, J. M. Zhang, F. Sarro, and M. Harman, "MAAT: a novel ensemble approach to addressing fairness and performance bugs for machine learning software," in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2022*, 2022, pp. 1122–1134.

[19] M. Hort, Z. Chen, J. M. Zhang, M. Harman, and F. Sarro, "Bias mitigation for machine learning classifiers: A comprehensive survey," *ACM Journal on Responsible Computing*, vol. 1, no. 2, pp. 1–52, 2024.

[20] S. Galhotra, Y. Brun, and A. Meliou, "Fairness testing: testing software for discrimination," in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2017, Paderborn, Germany, September 4-8, 2017*, E. Bodden, W. Schäfer, A. van Deursen, and A. Zisman, Eds. ACM, 2017, pp. 498–510. [Online]. Available: https://doi.org/10.1145/3106237.3106277

[21] S. Udeshi, P. Arora, and S. Chattopadhyay, "Automated directed fairness testing," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, 2018, pp. 98–108.

[22] A. Aggarwal, P. Lohia, S. Nagar, K. Dey, and D. Saha, "Automated test generation to detect individual discrimination in AI models," *CoRR*, vol. abs/1809.03260, 2018. [Online]. Available: http://arxiv.org/abs/1809.03260

[23] P. Zhang, J. Wang, J. Sun, G. Dong, X. Wang, X. Wang, J. S. Dong, and T. Dai, "White-box fairness testing through adversarial sampling," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, ser. ICSE '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 949–960. [Online]. Available: https://doi.org/10.1145/3377811.3380331

[24] H. Zheng, Z. Chen, T. Du, X. Zhang, Y. Cheng, S. Ji, J. Wang, Y. Yu, and J. Chen, "Neuronfair: Interpretable white-box fairness testing through biased neuron identification," in *Proceedings of the 44th International Conference on Software Engineering*, 2022, pp. 1519–1531.

[25] V. Monjezi, A. Trivedi, G. Tan, and S. Tizpaz-Niari, "Information-theoretic testing and debugging of fairness defects in deep neural networks," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023, pp. 1571–1582.

[26] T. Li, X. Xie, J. Wang, Q. Guo, A. Liu, L. Ma, and Y. Liu, "Faire: Repairing fairness of neural networks via neuron condition synthesis," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 1, pp. 1–24, 2023.

[27] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[28] M. Du, F. Yang, N. Zou, and X. Hu, "Fairness in deep learning: A computational perspective," *IEEE Intell. Syst.*, vol. 36, no. 4, pp. 25–34, 2021. [Online]. Available: https://doi.org/10.1109/MIS.2020.3000681

[29] T. M. Kodinariya, P. R. Makwana *et al.*, "Review on determining number of cluster in k-means clustering," *International Journal*, vol. 1, no. 6, pp. 90–95, 2013.

[30] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[31] T. V. Mele *et al.*, "COMPAS: A framework for computational research in architecture and structures." 2017-2021, http://compas.dev. [Online]. Available: https://doi.org/10.5281/zenodo.2594510

[32] P. Lahoti, A. Beutel, J. Chen, K. Lee, F. Prost, N. Thain, X. Wang, and E. H. Chi, "Fairness without demographics through adversarially reweighted learning," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/07fc15c9d169ee48573edd749d25945d-Abstract.html

[33] (2024) Website of this paper. [Online]. Available: https://sites.google.com/view/faireness-testing-grft

[34] Z. Wang, K. Qinami, I. C. Karakozis, K. Genova, P. Nair, K. Hata, and O. Russakovsky, "Towards fairness in visual recognition: Effective strategies for bias mitigation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 8919–8928.

[35] B. Sun, J. Sun, L. H. Pham, and T. Shi, "Causality-based neural network repair," in *44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022*. ACM, 2022, pp. 338–349. [Online]. Available: https://doi.org/10.1145/3510003.3510080

[36] G. Tao, W. Sun, T. Han, C. Fang, and X. Zhang, "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*, 2022, pp. 1173–1184.

[37] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[38] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[39] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 3730–3738.

[40] (2021) Stanford sentiment treebank dataset. [Online]. Available: http://nlpprogress.com/english/sentiment_analysis.html

[41] J. Chakraborty, S. Majumder, Z. Yu, and T. Menzies, "Fairway: a way to build fair ML software," in *ESEC/FSE '20: 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual Event, USA, November 8-13, 2020*, P. Devanbu, M. B. Cohen, and T. Zimmermann, Eds. ACM, 2020, pp. 654–665. [Online]. Available: https://doi.org/10.1145/3368089.3409697

[42] G. Nguyen, S. Biswas, and H. Rajan, "Fix fairness, don't ruin accuracy: Performance aware fairness repair using automl," in *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2023, San Francisco, CA, USA, December 3-9, 2023*, S. Chandra, K. Blincoe, and P. Tonella, Eds. ACM, 2023, pp. 502–514. [Online]. Available: https://doi.org/10.1145/3611643.3616257

[43] U. Gohar, S. Biswas, and H. Rajan, "Towards understanding fairness and its composition in ensemble machine learning," in *45th IEEE/ACM International Conference on Software Engineering, ICSE 2023, Melbourne, Australia, May 14-20, 2023*. IEEE, 2023, pp. 1533–1545. [Online]. Available: https://doi.org/10.1109/ICSE48619.2023.00133

[44] J. Wang, Y. Li, and C. Wang, "Synthesizing fair decision trees via iterative constraint solving," in *Computer Aided Verification - 34th International Conference, CAV 2022, Haifa, Israel, August 7-10, 2022, Proceedings, Part II*, ser. Lecture Notes in Computer Science, S. Shoham and Y. Vizel, Eds., vol. 13372. Springer, 2022, pp. 364–385. [Online]. Available: https://doi.org/10.1007/978-3-031-13188-2_18

[45] Z. Chen, X. Li, J. M. Zhang, F. Sarro, and Y. Liu, "Diversity drives fairness: Ensemble of higher order mutants for intersectional fairness of machine learning software," in *Proceedings of the 47th IEEE/ACM International Conference on Software Engineering, ICSE 2025*, 2025.

[46] Z. Chen, J. M. Zhang, F. Sarro, and M. Harman, "Fairness improvement with multiple protected attributes: How far are we?" in *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering, ICSE 2024*, 2024, pp. 160:1–160:13.

[47] Z. Chen, J. M. Zhang, F. Sarro, and M. Harman, "A comprehensive empirical study of bias mitigation methods for machine learning classifiers," *ACM Trans. Softw. Eng. Methodol.*, vol. 32, no. 4, pp. 106:1–106:30, 2023. [Online]. Available: https://doi.org/10.1145/3583561

[48] X. Li, Z. Chen, J. M. Zhang, Y. Lou, T. Li, W. Sun, Y. Liu, and X. Liu, "Benchmarking bias in large language models during role-playing," 2024. [Online]. Available: https://arxiv.org/abs/2411.00585

[49] T. Li, X. Zhang, C. Du, T. Pang, Q. Liu, Q. Guo, C. Shen, and Y. Liu, "Your large language model is secretly a fairness proponent and you should prompt it like one," 2024. [Online]. Available: https://arxiv.org/abs/2402.12150

[50] T. Li, Y. Cao, J. Zhang, S. Zhao, Y. Huang, A. Liu, Q. Guo, and Y. Liu, "Runner: Responsible unfair neuron repair for enhancing deep neural network fairness," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24. New York, NY, USA: Association for Computing Machinery, 2024. [Online]. Available: https://doi.org/10.1145/3597503.3623334

[51] T. Li, Q. Guo, A. Liu, M. Du, Z. Li, and Y. Liu, "FAIRER: Fairness as decision rationale alignment," in *Proceedings of the 40th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., vol. 202. PMLR, 23–29 Jul 2023, pp. 19 471–19 489. [Online]. Available: https://proceedings.mlr.press/v202/li23h.html

[52] T. Li, Z. Li, A. Li, M. Du, A. Liu, Q. Guo, G. Meng, and Y. Liu, "Fairness via group contribution matching," in *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, ser. IJCAI '23, 2023. [Online]. Available: https://doi.org/10.24963/ijcai.2023/49

[53] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. IEEE Computer Society, 2015, pp. 815–823. [Online]. Available: https://doi.org/10.1109/CVPR.2015.7298682