11-2023

Faire: Repairing fairness of neural networks via neuron condition synthesis

Tianlin LI

Xiaofei XIE
Singapore Management University, xfxie@smu.edu.sg

Jian WANG

Qing GUO

Aishan LIU

See next page for additional authors

Citation

LI, Tianlin; XIE, Xiaofei; WANG, Jian; GUO, Qing; LIU, Aishan; MA, Lei; and LIU, Yang. Faire: Repairing fairness of neural networks via neuron condition synthesis. (2023). *ACM Transactions on Software Engineering and Methodology*. 33, (1), 1-24.
Available at: https://ink.library.smu.edu.sg/sis_research/8475

Author

Tianlin LI, Xiaofei XIE, Jian WANG, Qing GUO, Aishan LIU, Lei MA, and Yang LIU

Faire: Repairing Fairness of Neural Networks via Neuron Condition Synthesis

TIANLIN LI, Nanyang Technological University, Singapore
XIAOFEI XIE, Singapore Management University, Singapore
JIAN WANG, Nanyang Technological University, Singapore
QING GUO, IHPC and CFAR, Agency for Science, Technology and Research, Singapore
AISHAN LIU, Beihang University, China
LEI MA, University of Alberta, Canada and The University of Tokyo, Japan
YANG LIU, Zhejiang Sci-Tech University, China, and Nanyang Technological University, Singapore

Deep Neural Networks (DNNs) have achieved tremendous success in many applications, while it has been demonstrated that DNNs can exhibit some undesirable behaviors on concerns such as robustness, privacy, and other trustworthiness issues. Among them, fairness (i.e., non-discrimination) is one important property, especially when they are applied to some sensitive applications (e.g., finance and employment). However, DNNs easily learn spurious correlations between protected attributes (e.g., age, gender, race) and the classification task and develop discriminatory behaviors if the training data is imbalanced. Such discriminatory decisions in sensitive applications would introduce severe social impacts. To expose potential discrimination problems in DNNs before putting them in use, some testing techniques have been proposed to identify the discriminatory instances (i.e., instances that show defined discrimination[1]). However, how to repair DNNs

[1] The formal definition of "discriminatory instances" is in Section 2.2.

This research/project is supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No. AISG2-PhD-2021-08-022T). This research/project is also partially supported by the National Research Foundation, Singapore, and the Cyber Security Agency under its National Cybersecurity R&D Programme (Grant No. NCRP25-P04-TAICeN), the Ministry of Education, Singapore under its Academic Research Tier 3 (Grant No. MOET32020-0004). It is also supported by A*STAR Centre for Frontier AI Research, the National Research Foundation Singapore and the National Research Foundation, Singapore under its the AI Singapore Programme (Grant No. AISG2-RP-2020-019), the National Research Foundation, Singapore, and DSO National Laboratories under the AI Singapore Programme (AISG Award No. AISG2-GC-2023-008). Lei Ma is supported in part by the Canada CIFAR AI Chairs Program, the Natural Sciences and Engineering Research Council of Canada (NSERC Grants No. RGPIN-2021-02549, No. RGPAS-2021-00034, and No. DGECR-2021-00019). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the Ministry of Education, Singapore.

Authors' addresses: T. Li and J. Wang, Nanyang Technological University, 50 Nanyang Ave, 639798, Singapore; e-mails: {tianlin001, jian004}@e.ntu.edu.sg; X. Xie (Corresponding author), Singapore Management University, 90 Stamford Rd, Singapore 17890; e-mail: xiaofei.xfxie@gmail.com; Q. Guo (Corresponding author), IHPC and CFAR, Agency for Science, Technology and Research, 1 Fusionopolis Way, #20-10, Connexis North Tower, Singapore 138632; e-mail: tsingqguo@ieee.org; A. Liu, Beihang University, 37 Xueyuan Road, Haidian District, Beijing, P.R. China, 100191; e-mail: liuaishan@buaa.edu.cn; L. Ma, University of Alberta, 116 St & 85 Ave, Edmonton, AB T6G 2R3, Canada and The University of Tokyo, Japan; e-mail: ma.lei@acm.org; Y. Liu, Zhejiang Sci-Tech University, China, and Nanyang Technological University, 50 Nanyang Ave, 639798, Singapore; e-mail: yangliu@ntu.edu.sg.

1049-331X/2023/11-ART21 $15.00
https://doi.org/10.1145/3617168

after detecting such discrimination is still challenging. Existing techniques mainly rely on retraining on a large number of discriminatory instances generated by testing methods, which requires huge time overhead and makes the repairing inefficient.

In this work, we propose the method *Faire* to effectively and efficiently repair the fairness issues of DNNs, without using additional data (e.g., discriminatory instances). Our basic idea is inspired by the traditional program repair method that synthesizes proper condition checking. To repair traditional programs, a typical method is to localize the program defects and repair the program logic by adding condition checking. Similarly, for DNNs, we try to understand the unfair logic and reformulate it with well-designed condition checking. In this article, we synthesize the condition that can reduce the effect of features relevant to the protected attributes in the DNN. Specifically, we first perform the neuron-based analysis and check the functionalities of neurons to identify neurons whose outputs could be regarded as features relevant to protected attributes and original tasks. Then a new condition layer is added after each hidden layer to penalize neurons that are accountable for the protected features (i.e., intermediate features relevant to protected attributes) and promote neurons that are accountable for the non-protected features (i.e., intermediate features relevant to original tasks). In sum, the repair rate[2] of *Faire* reaches up to more than 99%, which outperforms other methods, and the whole repairing process only takes no more than 340 s. The evaluation results demonstrate that our approach can effectively and efficiently repair the individual discriminatory instances of the target model.

CCS Concepts: • **Software and its engineering** → *Software testing and debugging*; • **Computing methodologies** → *Neural networks;*

Additional Key Words and Phrases: Deep learning repair, fairness, individual discrimination, model interpretation

ACM Reference format:
Tianlin Li, Xiaofei Xie, Jian Wang, Qing Guo, Aishan Liu, Lei Ma, and Yang Liu. 2023. Faire: Repairing Fairness of Neural Networks via Neuron Condition Synthesis. *ACM Trans. Softw. Eng. Methodol.* 33, 1, Article 21 (November 2023), 24 pages.
https://doi.org/10.1145/3617168

1 INTRODUCTION

Deep Learning (DL) has achieved tremendous success and demonstrated its great potential in solving complex tasks in many applications, such as image classification [16], speech recognition [44], and natural language processing [12]. However, **Deep Neural Networks (DNNs)** are known to be vulnerable and not reliable in terms of some properties, e.g., robustness, privacy, and safety [4, 9, 25]. Apart from these properties, fairness is also one key property, which may cause societal impact and is attracting more attention. For example, it has been shown that facial recognition technology can recognize white faces more easily than those from other backgrounds. Recent reports stated that some Uber Eats food delivery drivers were fired because facial recognition software could not recognize their faces [35]. Such bias has led some companies such as IBM and Microsoft to abandon their disputable technologies [34].

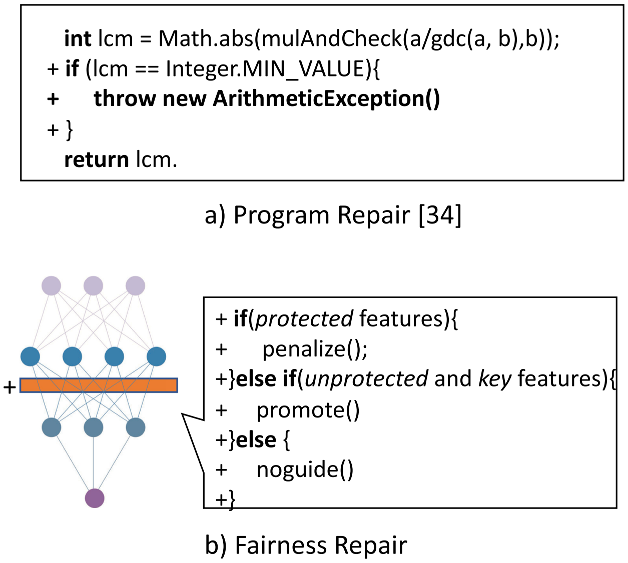
Non-discrimination has become one of the most critical factors for social protection and equal human rights [45]. However, recent works [58, 59] have shown that discrimination universally exists in machine learning models such as individual discrimination [15] and group discrimination [17]. Such discrimination is usually defined on some protected or sensitive attributes (e.g., gender, age, and race), and is easily triggered when training on an imbalanced dataset (i.e., strong correlations between target and protected attributes in this dataset) under a standard training process. Individual discrimination means that DNNs can make different predictions for individuals

[2] Repair rate is expressed as the ratio of the number of repaired instances and the number of all generated discriminatory instances to evaluate how effective *Faire* is in repairing individual discriminatory instances.

that only differ in the protected attributes (e.g., gender difference). Group discrimination means that DNNs have a bias in predicting different groups that also differ in the protected attributes. In fact, discrimination always exists and poses potential threats if the decision-making process of the model considers protected attributes as critical influencing factors. To mitigate such threats, in the software engineering community, a number of testing techniques [3, 45, 58, 59] have been proposed for detecting individual discriminatory instances in DNNs. When a large number of discriminatory instances have been discovered, a question naturally arises: how to repair the discrimination problem in DNNs?

Recently, although there have been some attempts on the DNN repair [23, 37, 42, 43, 50, 57], they mainly focus on repairing the robustness issues (i.e., incorrect predictions) and cannot be applied to repair fairness issues, because such methods only focus on the model prediction accuracy but ignore the fairness, which might even further deteriorate fairness of the original model. It is necessary and pressing to develop methods to repair the discrimination problem for given DNNs. Individual discrimination can identify discrimination behaviors that may be ignored by group fairness [19, 58]. Individual fairness enables effective/powerful/robust and granular verification of discriminatory behavior induced by changes only in the protected attribute in different contexts, whereas group fairness measures may not detect discrimination when the model exhibits opposite treatment to the same group under different situations. Therefore, in this work, we mainly focus on tackling the repairing problem in terms of individual discrimination in DNNs. One existing repairing method [48] is to design a loss function to remove protected attributes under the multi-task learning setting, while this method might fail to stably remove protected features. To repair individual discrimination, another intuitive idea is to remove the protected attributes before the training process. However, the individual discrimination may still exist due to the possible implicit correlations between the protected attributes and non-protected attributes [3] (e.g., the age attribute can be reflected by other properties such as occupation). However, some attributes (e.g., gender and race) are hard to remove in specific domains such as face recognition. Another typical repair method is to generate a large number of discriminatory instances and retrain the model by adding them to the training data. For example, the existing techniques AEQUITAS [45], ADF [59], and EIDG [58] usually have two steps: the *global search* is to generate diverse discriminatory instances and the *local search* aims to generate more analogous instances that will be used to retrain the model [58, 59] (More details will be introduced in Section 2.2). However, such a method has some drawbacks: (1) it relies on the generation of discriminatory instances, which needs a lot of time (e.g., hours to days), (2) retraining is also a time-consuming process due to substantial data augmentation, especially on large-scale tasks, and (3) the labels of the generated discriminatory instances are unknown. These factors significantly reduce the effectiveness of these techniques in practice, which is also confirmed by our evaluation results. Thus, an efficient, effective, and practical repair technique is required.

To this end, we propose a novel technique *Faire*, which can effectively and efficiently repair the discrimination of DNNs. Recall that the discrimination is caused because the prediction of the DNN is sensitive to the protected attributes. Our key insight is to reduce the effect of the protected attributes on the prediction. Specifically, inspired by the program repair technique [52] that synthesizes the conditions for patching the program, we try to infer the feature-level "condition" that will filter the features of protected attributes. Figure 1 shows one example that intuitively illustrates our basic idea. Figure 1(a) shows the program condition synthesized for capturing corner cases in traditional software. Similarly, we add one condition layer (the orange layer in Figure 1(b)) after each hidden layer of the DNN, which will penalize the protected features and promote the features that are not only non-protected but also accountable for the original tasks. For other features, we will not provide direct manipulation during the repair (i.e., we neither promote these nor



```
  int lcm = Math.abs(mulAndCheck(a/gdc(a, b),b));
+ if (lcm == Integer.MIN_VALUE){
+     throw new ArithmeticException()
+ }
  return lcm.
```

a) Program Repair [34]

```
+ if(protected features){
+     penalize();
+}else if(unprotected and key features){
+    promote()
+}else {
+    noguide()
+}
```

b) Fairness Repair

Fig. 1. Program Repair and Fairness Repair.

penalize these features). Specifically, we conduct neuron analysis to identify neurons that represent the protected features (protected neurons) and neurons that output features accountable for the prediction of the original task (non-protected neurons). Based on this, we will add the condition layer that penalizes the output values of the protected neurons and promotes the output values of the non-protected neurons at each layer. Finally, we fine-tune the model with the original training data by freezing the weights of the original hidden layers and only training the added condition layers. In this way, we can manipulate features of the hidden layers such that fairness could be enhanced. Note that our method does not need to retrain with a large number of discriminatory instances, because our method explicitly reduces the effect of the protected attributes on the prediction. Compared with the methods that require a large number of discriminatory instances to implicitly force the DNNs to learn how to reduce individual discrimination, our method is, therefore, more efficient. Moreover, different from the multitask learning method [48], our method adds a condition layer to control the protected features and the non-protected features, which is more effective and direct.

We have implemented *Faire* and evaluated it on widely used datasets. We compared *Faire* with three different baselines, i.e., the discriminatory instances retraining method, the flipping-based retraining method, and the multitask learning method. The experimental results show that, on average, *Faire* can successfully repair 3% more individual discriminatory instances than the best baseline method. We further evaluated the fairness of the repaired model using the state-of-the-art testing tools. The number of discriminatory instances detected by testing tools on our model is lower than that of the baseline models by 13% without dropping accuracy too much (less than 1%), indicating better fairness on our repaired model. Finally, we analyzed the efficiency of *Faire*. Compared with the discriminatory instances retraining method that achieves the best results among the baselines, our method is much more lightweight, i.e., tens of thousands of seconds less for the repair.

To summarize, this article makes the following contributions:

(1) We conduct an empirical study to assess the effect of features relevant to protected attributes on the individual discrimination. We reveal the correlation between protected features and

individual discrimination, and we are enlightened to manipulate features such that the fairness could be enhanced.

(2) We develop an effective and efficient method *Faire* to repair the individual discrimination of DNNs.
(3) We conduct a comprehensive evaluation on five popular tabular datasets and one image dataset. The results demonstrate the effectiveness and efficiency of our method.
(4) The source code, data, and more results are publicly available at our website [5].

To the best of our knowledge, this article is the first work to systematically analyze and repair the individual unfairness of DNNs, which does not need additional data. The main novelty is the lightweight repair based on the white-box neuron analysis. We design comprehensive experiments to show the effectiveness and efficiency of our methods. The results, especially the validation with the state-of-the-art testing tool, demonstrate that *Faire* can significantly improve the individual fairness of the target model. While program repair has been extensively studied in the **software engineering (SE)** community, the DNN repair in terms of different properties is less touched. This article provides a first step towards a very important problem, i.e., enhancing fairness that is widely concerned in the society. More importantly, unlike the methods of the AI community, our technology is motivated and designed from the perspective of SE technique (e.g., conditional synthesis), which can provide SE researchers with some guidance on the research of DNN repair.

2 BACKGROUND

In this section, we briefly introduce the relevant background including DNNs, individual discrimination, and problem definition.

2.1 Deep Neural Networks

DNNs are inspired by the neural networks of human brains, and they are popular due to excellent performance [24]. A DNN usually consists of multiple layers of neurons, and achieves meaningful information extraction and useless information discarding in a layer-wise manner [31].

Definition 2.1. A 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 $f : X \rightarrow Y$, 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 output values of neurons at l).

2.2 Individual Discrimination

Deep learning models are statistical models that are trained to maximize accuracy on the majority of examples, and they do so by exploiting the most discriminative cues in a dataset, potentially learning spurious correlations [38]. Models potentially cause severe fairness issues, if they capture spurious correlations between protected attributes and targets. However, the dataset collection process usually cannot guarantee a sufficient number of real-world samples that are under a balanced distribution. Meanwhile, even when datasets are balanced such that each label co-occurs equally with each protected attribute, current training algorithms may also amplify the association between labels and the protected attribute, as much as if the data had not been balanced [46]. Therefore, unfairness widely exists in deep learning models.

There are many different metrics evaluating the fairness of machine learning models [13]. Among these metrics, individual fairness follows the philosophy that similar inputs that only differ in some protected attributes should not yield discriminatory results. We inherit the definition of individual discrimination in Reference [59]. The attributes set of the dataset can be denoted by $A = A_1, A_2, \ldots, A_n$. Furthermore, we use $P \subset A$ to denote the protected attributes set such as gender, race, and age. Correspondingly, we use NP to denote the set of non-protected attributes. We define the discriminatory instance as below.

Definition 2.2 (Discriminatory Instance). $x = a_1, a_2, \ldots, a_n$ 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 $x' = a'_1, a'_2, \ldots, a'_n$ in the instance space that satisfies the following conditions:

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

As we can see from the definition, when x is a discriminatory instance, x' is also a discriminatory instance. As introduced in References [58, 59], the search of discriminatory instances is done round by round and an important indicator of individual fairness is how many discriminatory instances can be searched in given rounds. The discriminatory instance searching process consists of two parts: global generation and local generation. The goal of global generation is to detect diverse discriminatory instances via clustering. The local generation aims to generate more discriminatory instances nearby the instances detected by the global generation.

Specifically, in the global generation phase, Reference [59] clusters the samples in the original dataset and selects seed instances for each cluster in a round-robin fashion. Moreover, Reference [59] uses gradients to maximize the difference between the DNN outputs of two similar instances to guide the crafting of individual discriminatory instances. The selection process will continue until the number of generated instances reaches a certain number. The local generation then takes the instances generated by the global phase as input. To find more discriminatory instances, the local generation phase searches the neighbors of the input discriminatory instances. In this process, the gradients are used to generate instances that are minimally different from the seeds while maintaining their model predictions. Reference [58] improves Reference [59] both in global generation and local generation. In the global generation, they integrate the momentum term into the iterative search, which enables the memorization of the previous trend and identifies more discriminatory seeds. In the local phase, they designed a more direct strategy to select and perturb attributes to complement the strategy adopted in Reference [59].

2.3 Problem Definition

Our problem is 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 protected attributes as x', the classification output should be not changed, i.e., $f'(x) = f'(x')$. A stricter definition could be

$$\arg\min_{\theta} \sum_{l \in f^{\theta}} |f_l^{\theta}(x) - f_l^{\theta}(x')|, s.t., f^{\theta}(x) = f^{\theta}(x') \wedge f(x) = f^{\theta}(x),$$

where f^{θ} means the new model with the learned parameters θ, $f_l^{\theta}(x)$ represents features extracted at layer l of f^{θ} on input x. 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. To achieve this goal, it usually involves retraining on a large number of individual discriminatory instances and thus is not efficient. Without the availability of the individual discriminatory instances, our problem is transformed to reduce the effects of the protected attributes for prediction but still keep the normal functionality of the original model. Considering that the

Table 1. Correlation between Protected Attributes and Unprotected Attributes

Dataset	Removed attributes	Targeted attributes	Accuracy
Census Income	age, race, gender	age	0.603
		race	0.852
		gender	0.812
Bank Marketing	age	age	0.711
LSAC	gender, race	gender	0.594
		race	0.857

protected attributes may not be removed, in this work, we aim to reduce such effects in the feature space of the DNN.

3 METHODOLOGY

3.1 Motivation

Our work is motivated based on the following two observations:

(1) The discriminatory instances are not always available, since the generation of such instances is time-consuming and the generated data requires larger storage space, which is unrealistic in some scenarios. Moreover, the truth labels are unknown for the newly generated discriminatory instances.
(2) Directly removing the protected attributes cannot address the unfairness issue [3, 58], since there usually are strong correlations between protected attributes and unprotected attributes. Moreover, protected properties also cannot always be removed.

For the first observation, our evaluation results show that the generation usually consumes several days. A more detailed discussion can be found in our evaluation. For the second observation, to probe the correlations between protected and unprotected attributes, we remove protected attributes from training data and set the protected attributes as the classification labels. Table 1 shows the correlation estimation. We can observe that on the Census Income dataset, the race classification accuracy reaches 85.2%. On the LSAC and Bank Marketing dataset, the classification accuracies for protected attributes could also reach more than 80%. The results show that the non-protected attributes may have strong correlations with protected attributes. Hence, the protected attributes can still be inferred from unprotected attributes even if they are removed, i.e., the discrimination would still be implied in the non-protected attributes.

3.2 Overview

Motivated by the above observations, we propose a method *Faire* that aims to reduce the effects of the protected attributes when not explicitly removing them. Specifically, we analyze the features that are used for decision-making by the DNN. Then, we repair the model by reducing the impact of the features relevant to the protected attributes. In the following sections, we call the features of protected attributes and non-protected attributes as **protected features (*PF*)** and **non-protected features (*NPF*)**, respectively. The key challenges are how to identify the *PF* and remove the impact of *PF* while causing minimal loss on the accuracy of the model.

Figure 2 shows the overview of our method. Given an unfair DNN model f that conducts the original task (denoted as O-task), its working process can be considered as classifying the input x via the classifier component (i.e., the last several layers) on the ground of the features extracted by the backbone component (i.e., the first several hidden layers). We do neuron-level analysis regarding the features (F) to distinguish PF and NPF. To identify PF from F with regards to a

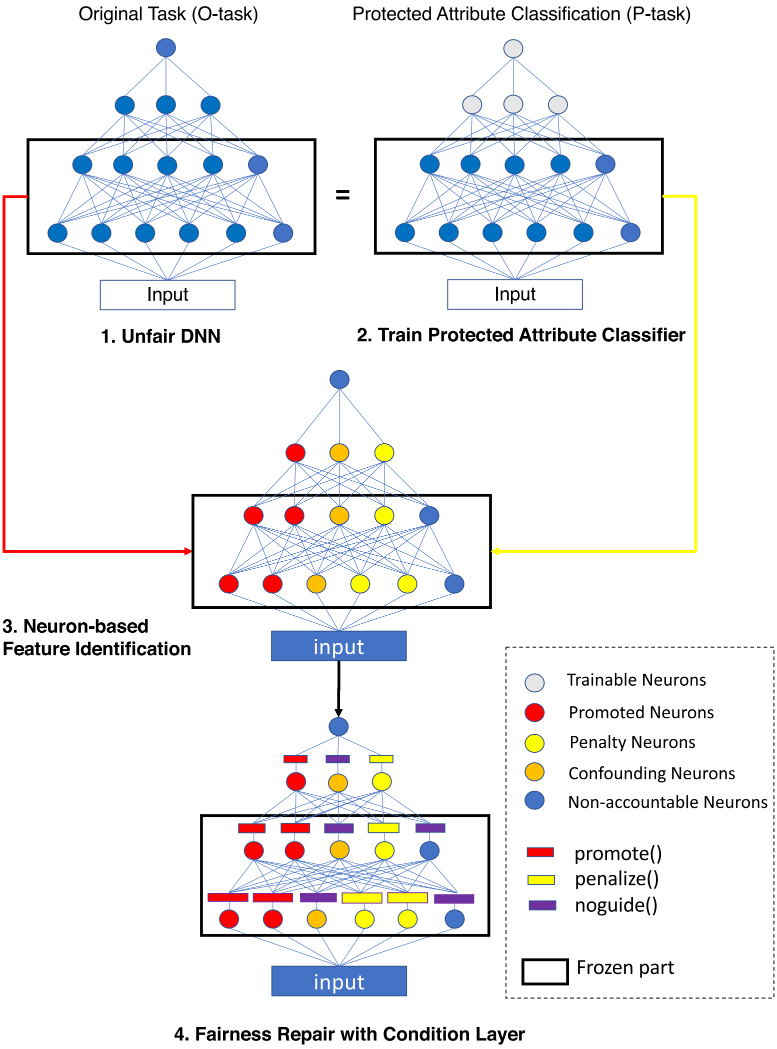


Fig. 2. Overview of this work.

protected attribute a, we base on the backbone component of the original model and train another model f' that recognizes the target protected attribute a (denoted as P-task). For example, if the protected attribute is gender, then the new model will classify whether this input is male or female. Note that, during the training of the new model, we *freeze* the weights of the backbone component of the original model and only set the last several layers as trainable. In this way, we can recognize the protected features, since f' uses more *PF* extracted by the original model. In Figure 2, the shared and frozen the backbone component in f and f' is marked in the black box.

Next, we will identify the key features that are used in f and f'. Specifically, we use the existing explanation technique [41] to identify the accountable neurons of each layer that contribute more to the decision of the given input. The outputs of these neurons can be used to represent the key features. Intuitively, the higher the contribution score of the neuron, the more accountable it is for the prediction. For example, in step 3, the red neurons represent the key features for the O-task and the yellow neurons represent the key features for the P-task. The orange neurons represent that they are accountable for both of P-task and O-task (called confounding neurons). The blue neurons represent that they are not accountable for both of P-task and O-task. Specifically, the model f can work well on O-task due to the features of red neurons. The discrimination exists in f, because the features of yellow neurons also have some effects on the prediction. Thus, our key idea of fairness repair is to *promote the features of red neurons and penalize the features of yellow neurons*. The promotion of red neurons is to ensure the functionality of the original model, while the penalty of yellow neurons is to eliminate discrimination.

Finally, we propose to add a new layer (called condition layer) after each hidden layer. The condition layer will promote the features of red neurons and penalize the features of yellow neurons (see Step 4 in Figure 2 and Figure 1). For other neurons such as confounding neurons and non-accountable neurons, we will not provide specialized manipulation methods. Specifically, the condition layer controls the data flow of the DNN by multiplying the output value of neurons with the learnable parameters. We freeze the weights of the backbone component and only train the added condition layers as well as the classifier component. Our method is similar to the basic idea of traditional program repair, i.e., we do not retrain a totally new model but only synthesize some suitable patch layers in the original model.

3.3 Study on Protected Features

In this section, we conduct an empirical study to demonstrate the connection between *PF* and the final prediction fairness. We here compare *PF* in the vanilla models and the fairer models (i.e., repaired models). To estimate the *PF* used in the model, we train protected attribute classifiers as a proxy. The higher the classification accuracy of protected attributes, the more *PF* is used in the model. We first relabel the original training data. For example, in Census Income, the original task is income prediction and there is one protected attribute (i.e., gender) in the training data. We relabel the training data with gender labels (i.e., man and woman). We reuse and freeze the first several layers of the original model, and we retrain the rest part of the model to recognize new labels. For example, "a:repaired model" is trained by the following process: freeze the first several layers of the repaired model and retrain the rest part for the classification of "a" attribute. A higher accuracy of the retrained model means that more *PFs* tend to be provided by the first several layers in the corresponding original model. The accuracy of "a:vanilla model" is higher than that of "a:repaired model," we can say that more PFs relevant to attribute "a" tend to be provided by the first several layers in the corresponding original model (i.e., the vanilla model).

Then, we conduct a study to demonstrate our basic assumption: *the PF used in the model can have a large impact on its fairness.* To this end, for comparison, we use the original model (f_0) that is unfair and a fairer model (f_1) that is retrained by the state-of-the-art method [58]. In this method, EIDIG searches diverse discriminatory instances in the global phase, and then they further develop the local generation for generating more individual discriminatory instances to retrain (details are introduced in Section 4.1.2, Discriminatory instances retraining). Then, we train the corresponding new models (denoted as f_0' and f_1') for recognizing the protected attributes. The hypothesis is that *the accuracy of f_0' is generally higher than f_1'*. If it is verified, then it means that more *PF* are extracted in the frozen layers from the unfair model f_0, i.e., using more *PF* could harm fairness.

Figure 3 shows the results on three datasets (i.e., Census, Bank, and LSAC, more details about these datasets are deferred to Section 4.1). Note that we freeze different numbers of layers in the DNN. *freeze i* represents that we freeze the first *i* layers of the model. For each attribute, we show two lines representing the accuracy of f_0' and f_1', which are retrained based on the vanilla model and the repaired model. The results clearly show that the line of f_0' is above the line of f_1', i.e., f_0' extracts more *PF*, which means f_0 utilizes more *PF* for classification than f_1. The results confirmed

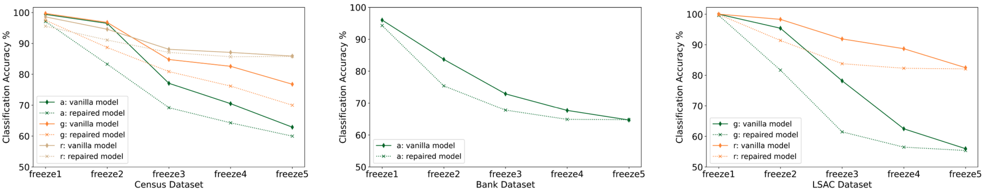


Fig. 3. Protected attributes classification performance on three tabular datasets.

our assumption. We also observed that, when more layers are frozen, it is usually harder to train a high-performance classifier in terms of the P-task, which tends to show that *PF* is potentially discarded layer by layer.

In summary, the results of the study could show that the *PF* used in the model is the root cause of the unfairness, which motivates us to reduce its impact in our method. Therefore, our objective is converted to minimize the impact of the protected attributes on the internal features so that the classification results would be less influenced by the protected attributes (i.e., smaller distance between internal features of inputs that only differ in protected attributes). Inspired by condition synthesis, we aim to synthesize the condition that can reduce the effect of protected features in the DNNs. We propose to analyze the neuron-based features first and add a condition layer to control the features. Then our objective is to learn the weights of the condition layer to repair the individual discrimination.

3.4 Neuron-based Feature Analysis

In this section, we will introduce how to extract the relevant features. In a deep neural network, the neurons are assumed to have different functionalities for the task. Given an input x, we can analyze the neuron outputs to understand the neuron functionalities for the task. We adopt the existing explanation technique (e.g., DeepLIFT [41][3]) to analyze the contribution of each neuron for the classification task. Specifically, the higher the contribution score, the greater the impact on the prediction. The neurons with higher contribution can represent the key features of the prediction. We name neurons with higher contribution as accountable neurons, and the definition of accountable neurons for one single instance is as follows:

Definition 3.1 (Instance-level Accountable Neurons). Given an input x and the DNN model f, we define its **Accountable Neurons (ANs)** of f as a sequence of neurons sets $p = \langle s_0, \ldots, s_L \rangle$, where s_l is a set of accountable neurons that have the largest contribution scores at the layer l and

$$\forall n \in s_l, C_l^n \in \text{top-}k(C_l),$$

where C_l is the contributions scores of all neurons at layer l, top-$k(\cdot)$ is the function that extracts the top k percent contribution scores from C_l, and C_l^n represents the contribution score of the neuron n at the layer l. The parameter k controls the number of selected accountable neurons at each layer.

Based on the accountability analysis on the instance level, we then analyze the accountable neurons for the overall decision-making of the DNN. We calculate the accountable neurons for each training data and then select the common neurons that are accountable for most of the training data. For each training data $x \in X$, we calculate its instance-level accountable neurons at the layer

[3] The used interpretation techniques are orthogonal to our method. Compared with other interpretation techniques, DeepLIFT could handle cases where other techniques may give misleading results [41]. Moreover, it is easier to implement and integrate DeepLIFT to analyze the neuron function.

l. Then, we calculate a weight for a neuron n of layer l, which measures its accountability for the overall training data as follows:

$$w_n = \frac{|\{x | x \in X \wedge n \in s_l^x\}|}{|X|},$$

where s_l^x represents the instance-level accountable neurons at layer l for the instance x, $|X|$ is the total number of the data X.

Intuitively, the higher the weight, the more accountable the neuron n is for the decision of the DNN. We finally identify the accountable neurons of the DNN that extract the key features for the prediction:

$$\hat{p} = \langle \hat{s}_1, \ldots, \hat{s}_L \rangle,$$

where $\hat{s}_l = \{n | n \in l \wedge w_n > t\}$ and t is a threshold to control the neuron selection based on the weights.

3.5 Neuron Condition Analysis

Based on the neuron-based analysis, we calculate the accountable neurons for both the original classifier f and the protected attribute classifier f', denoted as $\hat{p_f}$ and $\hat{p_{f'}}$.[4] To improve the fairness, we divide the neurons of the DNN into four categories:

(1) *Penalized Neurons.* At layer l, the penalized neurons are defined as $\hat{s'}_l \setminus \hat{s}_l$, where $\hat{s'}_l$ and $\hat{s}_l$ are the accountable neurons calculated from f' and f, respectively. Intuitively, the neurons, which are accountable for the protected attribute classification but not accountable for the original classification, should have less impact. Thus, we should penalize the outputs of these neurons. We do not penalize all neurons in $\hat{s'}_l$, since some neurons may largely affect the performance of the original model.

(2) *Promoted Neurons.* On the contrary, the promoted neurons at layer l are defined as $\hat{s}_l \setminus \hat{s'}_l$, which are accountable for the original task but not accountable for recognizing the protected attributes. Thus, the prediction should depend more on these neurons, and we will promote their output.

(3) *Confounding Neurons.* There are some neurons that exist in both $\hat{s'}_l$ and $\hat{s}_l$, i.e., $\hat{s'}_l \cap \hat{s}_l$. These neurons could play important roles in both tasks. Thus, we cannot simply penalize them or promote them.

(4) *Non-accountable Neurons.* There are also some neurons that do not contribute much for both two tasks, i.e., $\{n | n \in l \wedge n \notin \hat{s'}_l \wedge n \notin \hat{s}_l\}$. Similarly, it is unclear whether such neurons should be promoted or penalized. However, these neurons tend not to affect the result too much.

3.6 Fairness Repair with Condition Layer

Based on the functionality analysis of different neurons (see Section 3.5), we locate *PF* and *NPF*, and then propose a method to repair the model without changing the original weights. Specifically, we define a condition layer that will be added after each frozen layer. The condition layer will perform different operations on the features. The condition layer c after the hidden layer l is a linear function that is defined as

$$c^{(l)}(f_l(x)) = W^{(l)} \cdot f_l(x),$$

where W is the learnable weights in condition layer and $f_l(x)$ is the neuron output of the layer l.

[4] Without loss of generality, our method can be extended on multiple protected attributes by training multiple protected attribute classifiers.

Intuitively, each neuron output will be multiplied by one parameter (i.e., the weights of the condition layer). By adjusting the weights, we can control the features used for the prediction such that the model is repaired to be fair. At first, we set the default weights of each condition layer as 1, which means that it is the same as the original model. Then, we train the weights of the condition layer on the original training data with a new loss. Our basic idea is to penalize the output of the *penalized neurons* while promoting the output of the *promoted neurons*. For the *confounding* and *non-accountable* neurons, we will not explicitly manipulate their outputs. Our loss function $\mathcal{L}$ contains two components: the original classification loss and the condition loss, which are defined as follows:

$$\mathcal{L}_c(x, y; \theta) = -(ylog(f^{\theta}(x)) + (1-y)log(1-f^{\theta}(x))), \tag{1}$$

$$\mathcal{L}_w(x; \theta) = \sum_{l}^{l \in f} \left(\sum_{n \in Pen_l} \mathbf{w}_l^n - \sum_{m \in Pro_l} \mathbf{w}_l^m \right), \tag{2}$$

$$\mathcal{L} = \mathcal{L}_c(x, y; \theta) + \lambda \mathcal{L}_w(x; \theta), \tag{3}$$

where $\mathcal{L}_c$ is the original cross entropy loss and $\mathcal{L}_w$ is the condition loss. Pro_l and Pen_l denote the promoted neurons and the penalized neurons at layer l, respectively. w_l^n denotes the weight for the neuron n in the condition layer and λ is used to balance these two loss terms. For the condition loss $\mathcal{L}_w$, we aim to increase the weights of the promoted neurons and decrease the weights of the penalized neurons. Intuitively, the larger weights mean that more features of the corresponding neurons are used while smaller weights can reduce the influence of the features. To maintain the accuracy of the target model, we add the cross entropy loss item $\mathcal{L}_c$ into our loss design. Through the loss Equation (3), the model will be trained to increase and decrease the corresponding weights (the aim of the second loss item $\mathcal{L}_w$) while maintaining the classification accuracy (the aim of the first loss item $\mathcal{L}_c$). To enable the training to converge, we define the **lower boundary (*lb*)** and the **upper boundary (*ub*)**, which are the minimum and maximum of the weights.

Note that during training, we freeze the backbone component (i.e., the first several layers) of the original model (see Figure 2) and fine-tune the classifier component (i.e., the last two layers). The fine-tuning process is less time-consuming than retraining from scratch.

4 EVALUATION

To evaluate the effectiveness and efficiency of our approach, we have implemented *Faire* based on Keras [11] with the Tensorflow backend [1]. The source code and more experimental details can be found on our website [5]. Specifically, our evaluation aims to answer the following research questions:

- **RQ1:** How effective is *Faire* in repairing individual discriminatory instances?
- **RQ2:** How effective is *Faire* in improving the individual fairness of the DNN?
- **RQ3:** Why could *Faire* outperform other methods?
- **RQ4:** How efficient is *Faire* in repairing fairness?

4.1 Experimental Setup

4.1.1 Datasets and Models. We evaluate *Faire* on four popular datasets including Census Income [14], Bank Marketing [14], LSAC [26], German Credit [14], COMPAS [33], and MNIST [27], where the first five datasets are in tabular form and the last one belongs to the image domain. These tabular datasets are commonly used in individual fairness testing [45, 58, 59].

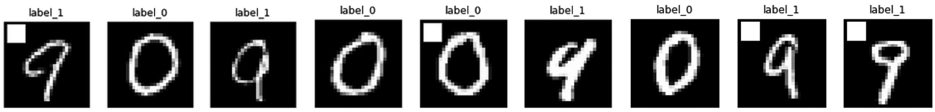


Fig. 4. Images on the MNIST dataset with white boxes and black boxes in the top left corner. The black boxes are not explicitly visible due to the black background.

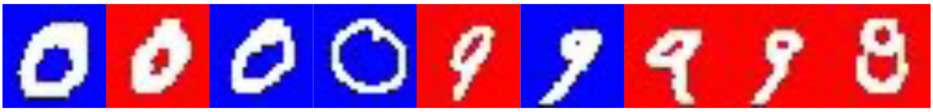


Fig. 5. Images on the Colored MNIST dataset with different colors.

- **Census Income.** The dataset was done by Barry Becker from the 1994 Census database with 48,842 instances and 14 attributes. The original aim is to determine whether a person makes over 50K a year. Amid these 14 attributes, race, gender, and age are defined as protected attributes.
- **Bank Marketing.** The dataset has more than 45,000 instances and 16 attributes of which the only protected attribute is age. The original aim of this dataset is to assess if the product (bank term deposit) would be ("yes") or not ("no") subscribed.
- **LSAC.** LSAC is a dataset that tracks students who entered law school in the fall 1991 through three or more years of law examinations. National race- and gender-specific bar passage data have been available for analysis and study.
- **German Credit.** This dataset is to give an assessment of credit based on personal and financial records. The small dataset has 1,000 examples with 20 attributes.
- **COMPAS. Correctional Offender Management Profiling for Alternative Sanctions (COMPAS)** is a well-known commercial algorithm that judges and parole authorities use to determine whether a criminal defendant is likely to commit another crime (recidivism). Based on 2-year follow-up research, it has been demonstrated that the algorithm is biased against black inmates and in favor of white defendants (i.e., those who committed crimes or violent crimes after 2 years).
- **MNIST.** The MNIST database of handwritten digits has a training set of 60,000 examples and a test set of 10,000 examples. We select this dataset to evaluate the usefulness of our method in the image domain. Note that it is hard to measure the individual discrimination in the image domain as the protected features could not be changed directly (see Definition 2.2). Thus, we add a box in the top left corner of the image as shown in Figure 4, and use different colors of the box (i.e., white and black) to simulate the bias. We pick data of two labels "0" and "9" from the MNIST dataset as our training data for binary classification. We set 20% of "0" samples as with white boxes and 80% of "9" samples as with white boxes. We assume the box color is the protected attribute. To evaluate individual discrimination, for instance, we change its box color to its counterpart, which contains the opposite protected information. If the predictions of the original instance and its counterpart are inconsistent, then the instance is a discriminatory instance. We also follow Reference [6] to construct the Colored Mnist dataset. We pick data of two labels "0" and "9" from the MNIST dataset as our training data for binary classification. We set 80% of "0" samples as blue and 80% of "9" samples as red. More details are shown in Figure 5.

Table 2. Accuracy and Architecture of Our Vanilla Models

	acc	$l1$	$l2$	$l3$	$l4$	$l5$	$l6$
Census	0.847	30	20	15	15	10	1
Bank	0.892	30	20	15	10	5	1
LSAC	0.863	50	30	15	10	5	1

For the tabular datasets, we adopt the toolkit of EIDIG [58] to process the datasets by converting all attributes to categorical ones. We follow the setting [58, 59] and train the six-layer fully connected DNN models. The detailed architecture is shown in Table 2.

4.1.2 Baselines. To evaluate the effectiveness of *Faire*, we select three baselines for comparison.

- **Discriminatory instances retraining** [58, 59]. In this method, ADF and EIDIG search diverse discriminatory instances in *global phase*, and then they further develop the *local generation* for generating more individual discriminatory instances to retrain. As introduced in Reference [58], EIDIG achieves better retraining performance. Thus, in this article, we mainly use EIDIG to generate data for retraining. More specifically, the default retraining strategy introduced in Reference [58] is to integrate discriminatory instances generated for all possible protected attribute combinations into training data to retrain. We here denote models retrained by this default strategy as M_{dis}). We also adopt a more reasonable strategy. When repairing the fairness problem relevant to a specific protected attribute, we only include discriminatory instances with regards to this protected attribute rather than all protected ones into training data (denoted as M_a).
- **Flipping-based retraining.** A basic method is to augment the training data by flipping the protected attributes of each training data (e.g., generate a new instance by only changing the gender from woman to man). During the learning process, the model will be learned to be insensitive to the protected attributes, because the ground truth stays unchanged when only the protected attributes vary in the training data. We denote this baseline as M_{flip}.
- **Multitask Learning.** Recall that our key insight is to eliminate the protected features in the original model. A popular method is to reduce these features with a multi-task setting [48]. Similar to our method, during the training of the original task, we can train the protected attribute classifier that shares the same backbone (i.e., the same hidden layers) at the same time. Then the loss function for updating the weights is to minimize the original cross-entropy loss ($\mathcal{L}_1$) and maximize the cross-entropy loss ($\mathcal{L}_2$) of the protected-attribute classifier. The final loss is $\mathcal{L} = \lambda_1\mathcal{L}_1 - \lambda_2\mathcal{L}_2$. In this way, the learned classifier could remove protected features. We denote this method as M_{mt}.

4.1.3 Evaluation Metrics. We design the following metrics for our evaluation:

- **Accuracy (*ACC*).** The accuracy of the repaired model is an important indicator for measuring its utility. The objective of *Faire* is to improve the individual fairness of the given models without sacrificing too much accuracy.
- **Repair Rate (*RR*).** Following the same setting in ADF and EIDIG, we measure the repair rate of each method. Specifically, given a set of individual discriminatory instances of the original model, we then calculate how many original discriminatory instances do not suffer from discrimination in the newly repaired model. Repair rate is expressed as the ratio of the number of repaired instances and the number of all generated discriminatory instances.
- **Detection Success Rate.** Only measuring the repair rate on the existing dataset is not enough, since a higher repair rate does not indicate that the new model is fairer. To

evaluate the fairness of the repaired model, we adopted the testing technique [58, 59] to generate the individual discriminatory instances in a fixed number of iterations. Specifically, we adopt two strategies that are used in References [58, 59] to evaluate the fairness: (1) given a fixed number of initial seed inputs, we adopt the *global search* of ADF or EIDIG to generate individual discriminatory instances for each seed. Then, we calculate the success rate, i.e., how many seeds based on which the testing tool can successfully generate discriminatory instances (denoted as *GS*). (2) For the random generation, we generate a fixed number of inputs by randomly combining the values of input attributes as introduced in EIDIG, and check the ratio of discriminatory instances in the inputs (denoted as *RG*).

4.2 Configuration

We mainly use EIDIG to generate a set of discriminatory instances d_{att} for a specific combination of protected attributes att. For the baseline M_{dis}, we follow the same setting in Reference [58] by merging all discriminatory instances (for different combinations of protected attributes) under three generation settings (i.e., *ADF*, $EIDIG_5$, $EIDIG_{INF}$) and selecting 5% of them for the retraining. For M_a, we retrain the model with the specific discriminatory instances d_{att} generated for the corresponding protected attribute(s) att. For M_{flip}, given a training instance, we replace $attr$ values with all possible values to generate new instances. We generate new instances for all training data and the labels of new instances are the same as the original training instances. We then integrate all these newly generated instances into training data for retraining. For M_{mt}, we here set both λ_1 and λ_2 as 1.0. Then the final loss is $\mathcal{L} = \mathcal{L}_1 - \mathcal{L}_2$. For *Faire*, λ in Equation (3) is set as 1.0. We freeze the first four layers as the backbone component and fine-tune the last two layers. We set 10 epochs for fine-tuning that can converge in these tasks. For *Faire*, we choose the last two layers to perform the classification task. Because we find that when only the last layer is selected to perform the classification task, the accuracy performance would degrade severely and this is against our objective. When more layers are chosen to perform the classification task, it is harder to eliminate protected features, because more PFs are in lower layers and the fairness improvement is limited.

All these experiments are conducted on the Intel Xeon Silver 4214 Processor with 1 Tesla V100 GPUs with 16 GB memory.

4.3 RQ1: Repair Rate

We evaluate the repair rate on the generated discriminatory instances set d_{att} for att. For *Faire*, we set k (see Definition 3.1) and t (the selection rate for $\hat{s}_l$) as 0.3 and 0.2, respectively. We set lb and ub as range $[-1, 0)$ with an interval 0.05, and range $[0, 1)$ with an interval 0.05, respectively.

Table 3 shows the results of different methods on different protected attributes, where *Attr* shows the protected attributes that are selected based on the setting in ADF and EIDIG. *ACC* and *RR* show the accuracy and the repair rate of the model repaired by each method, and M_{van} represents the original model. lb and ub show the optimal parameters that achieve the best results. Note that we also evaluate the parameters about k and t. Note that the symbol "—" means that the existing testing techniques [58, 59] cannot be directly used to generate individual discriminatory instances in the image domain. The complete results with different parameters (i.e., k, t, lb, and ub) and the analysis could be found in our website [5].

From the results, we can observe that *Faire* has a much higher repair rate than all baselines on all $attr$. In addition, the repair rate on a single attribute is higher than the repair rate on the combination of two attributes, indicating that fairness repair on the combination of protected attributes is more difficult. For model M_a, we can observe that the repair rate is mostly higher than that of M_{dis}. For example, in the LSAC dataset, the repair rates of attribute g and $g\&r$ of M_a (0.963, 0.951) are higher than those of M_{dis} (0.918, 0.878). It is because the discriminatory

Table 3. Repair Rate of Our Repaired Model

Dataset	Attr	M_{van}	M_{dis}		M_a		M_{mt}		M_{flip}		*Faire*			
		ACC	ACC	RR	ACC	RR	ACC	RR	ACC	RR	ACC	RR	*lb*	*ub*
Census	a	0.847	0.841	0.941	0.838	0.968	**0.846**	0.435	**0.846**	0.451	0.831	**0.999**	−0.30	0.15
	g	0.847	0.841	0.976	0.840	0.948	**0.849**	0.710	0.845	0.856	0.837	**0.995**	−0.60	0.10
	r	0.847	0.841	0.978	0.844	0.955	0.842	0.853	**0.846**	0.767	0.839	**0.994**	−0.95	0.80
	a&g	0.847	0.841	0.908	0.843	0.962	0.844	0.347	**0.845**	0.293	0.833	**0.990**	−0.30	0.25
	a&r	0.847	0.841	0.924	0.841	0.958	**0.847**	0.369	0.845	0.433	0.832	**0.987**	−0.35	0.25
	r&g	0.847	0.841	0.957	0.842	0.942	**0.846**	0.382	0.843	0.545	0.835	**0.972**	−0.90	0.80
Bank	a	0.892	0.890	0.916	0.890	0.977	**0.892**	0.838	0.889	0.478	0.890	**0.998**	−0.15	0.30
LSAC	g	0.863	0.859	0.964	**0.860**	0.947	**0.860**	0.914	0.857	0.679	0.832	**0.997**	−0.85	0.20
	r	0.863	0.859	0.918	0.848	0.963	0.858	0.884	**0.861**	0.806	0.833	**0.998**	−0.90	0.05
	g&r	0.863	0.859	0.878	0.844	0.951	**0.861**	0.841	0.850	0.448	0.830	**0.993**	−0.55	0.45
MNIST	box	0.995	—	—	—	—	**0.995**	0.333	**0.995**	0.500	0.993	**1.000**	−0.05	0.05
Credit	a	0.782	0.745	0.951	0.758	0.943	**0.762**	0.807	0.752	0.627	0.743	**0.958**	−0.50	0.60
	g	0.782	0.745	0.941	**0.772**	0.955	0.752	0.820	0.748	0.728	0.765	**0.997**	−0.10	0.60
	a&g	0.782	0.745	**0.991**	**0.762**	0.988	0.752	0.719	0.730	0.677	0.743	0.913	−1.00	0.25
COMPAS	g	0.675	0.668	0.733	0.658	0.703	**0.677**	0.653	0.657	0.270	0.637	**0.995**	−0.10	0.40
	r	0.675	0.668	0.883	**0.676**	0.859	0.673	0.787	0.661	0.871	0.659	**0.997**	−0.10	0.50
	g&r	0.675	0.668	0.672	0.671	0.642	0.672	0.481	**0.675**	0.128	0.645	**0.953**	−0.10	0.45

We run five times with different seeds and the average results are reported.

instance set d_{att} focuses more on discrimination relevant to att. Retrained with d_{att}, M_a will own a stronger capability to filter information relevant to att to maintain fair decision, while M_{dis} is not only targeting on att. The repair of M_{mt} is erratic (i.e., the range is 0.567). We observe that in the training process, the outputs of the adversary branch may verge to one fixed value instead of making the opposite predictions for protected attributes. This phenomenon reveals that this multitask learning paradigm is not stable enough. For M_{flip}, the repair rate is also unstable (the repair rate fluctuates from 0.293 to 0.856). In this retraining manner, M_{flip} can learn to ignore protected attributes only on the training data. However, the change of the unprotected attributes may still affect the sensitivity of the protected attributes in the prediction.

As for the accuracy, we can see that all repaired models can decrease the accuracy a bit, which indicates that the protected attributes tend to have a influence on the tasks. Compared with others, our accuracy has dropped most but not too much (less than 1%). We conjecture two possible reasons: (1) *Faire* makes a drastic change in the original neural network (i.e., add new condition layers) and fine-tunes the new model for a few epochs. (2) There is a trade-off between accuracy and fairness [30] and *Faire* indeed limits the information given to decision-makers. Our method achieved the best results in fairness repairing while sacrificing accuracy. For models trained by flipping-based methods (e.g., M_{flip}) and multitask learning (e.g., M_{mt}), their accuracies decrease less. We speculate that these two methods are closer to the original training with similar loss and similar training data. Differently, M_{dis} and M_a change the training data a lot (i.e., add more discriminatory instances), while *Faire* directly patches the DNN, leading to a decrease of the accuracy. We also conduct experiments on the Colored Mnist dataset, and we find that even though our method could effectively improve the repair rate (e.g., our method could achieve the RR as 0.689), the repair is not as effective as on the tabular dataset. The potential reason might be that some features, such as color discrimination, may be more difficult to locate and repair through an additional condition layer. This is because these features may be distributed across many neurons in a complex network, making it challenging to identify which specific neurons are responsible for them. Additionally, the nature of these features may make it more difficult to manipulate or adjust them without affecting other aspects of neural processing.

Considering the datasets in different domains, *Faire* is more generally applicable to different domains while other techniques are often difficult to be used in non-tabular datasets (e.g., image). This is because they are based on augmenting data by changing the values of (un)protected attributes. However, the value change on non-tabular datasets is not easy. Differently, *Faire* does not change the data directly and trains another neural network that works well for both tabular and non-tabular datasets.

> **Answer to RQ1:** *Faire* achieved a much higher repair rate (91.3%–100.0%) that largely outperforms all baselines on protected attributes of all datasets. In addition, *Faire* could also be applicable to image datasets while other techniques relying on data augmentation cannot. However, *Faire* drops more on the accuracy, indicating that there tends to be a trade-off between accuracy and fairness.

4.4 RQ2: Effectiveness of Fairness Repair

Setup. For the repaired models trained in *RQ1*, we then evaluate their fairness with the state-of-the-art fairness testing tool EIDIG and ADF. Specifically, for global search generation, we randomly select 1,000 seed inputs from the training data to calculate the success rate (i.e., *GS*). We follow the setting [58] and set the cluster number to 4 for ADF and EIDIG and the decay factor to 0.5 for EIDIG. We repeat this process 5 times to mitigate randomness. For then random generation, we randomly generate 1,000 inputs to calculate the ratio of discriminatory instances (i.e., *RG*). We repeat 10 times to calculate the average results.

Results. Table 4 shows the results of different methods on the five datasets. The results demonstrate that *Faire* significantly outperforms the existing methods. For the random generation *RG*, *Faire* is clearly better than other methods. For example, in the first three datasets (i.e., Census, Bank, and LSAC), the average ratios of discriminatory instances generated on our repaired models are only 0.004, 0.002, and 0.006, while the best results of other methods are 0.025 (M_{dis}), 0.011 (M_a), and 0.016 (M_a), respectively.

Moreover, we also use the fairness testing tool to generate discriminatory instances on the selected seeds. Columns GS_A and GS_E represent the ratio of discriminatory instances searched by ADF and EIDIG, respectively. The results show that the state-of-the-art tools ADF and EIDIG discovered fewer discriminatory instances on our repaired model. For example, EIDIG only achieved discriminatory ratios of 0.022, 0.003, and 0.018 on the first three datasets, respectively. For the retraining-based method, we found that M_a can achieve better fairness than M_{dis}, indicating that retraining with more discriminatory instances is not necessarily helpful for improving fairness. For example, in Bank, with ADF, the success rate on the model retrained with all data is 0.407 but it is 0.093 when retraining with only attribute-wise discriminatory instances. The naive flipping-based method M_{flip} obtains the worst performance (e.g., 0.896 and 0.869 in the Bank dataset), which indicates that the simple flipping of the protected attributes is ineffective for improving the fairness. For M_{mt} that is similar to *Faire* (i.e., does not rely on other data), the results are better than M_{flip} but worse than the retraining-based methods M_{dis} and M_a. Considering that M_{mt} and *Faire* share a similar basic idea (i.e., reduce protected features in the model), the results could show that our neuron-based analysis can better reduce the impact of *PF* than the fully learn-able method in M_{mt}.

> **Answer to RQ2:** *Faire* significantly outperforms other methods in improving the fairness of models. Although other methods could repair many original discriminatory instances (in RQ1), the fairness of new models may not be really enhanced.

Table 4. Average Results of Fairness Testing on Repaired Models

Data	M_{van}			M_{dis}			M_a			M_{mt}			M_{flip}			*Faire*		
	GS_A	GS_E	RG	GS_A	GS_E	RG	GS_A	GS_E	RG	GS_A	GS_E	RG	GS_A	GS_E	RG	GS_A	GS_E	RG
C-a	0.464	0.654	0.111	0.305	0.218	0.023	0.220	0.132	0.016	0.540	0.749	0.183	0.736	0.743	0.176	**0.001**	**0.001**	**0.001**
C-g	0.187	0.282	0.039	0.105	0.070	0.016	0.353	0.255	0.064	0.252	0.398	0.054	0.262	0.301	0.034	**0.006**	**0.010**	**0.000**
C-r	0.203	0.324	0.105	0.168	0.122	0.014	0.281	0.194	0.030	0.200	0.323	0.077	0.378	0.460	0.098	**0.013**	**0.017**	**0.003**
C-a&g	0.518	0.717	0.151	0.360	0.279	0.031	0.161	0.074	0.013	0.594	0.788	0.285	0.828	0.822	0.298	**0.028**	**0.034**	**0.003**
C-a&r	0.608	0.740	0.211	0.413	0.339	0.036	0.121	0.087	0.009	0.616	0.802	0.272	0.758	0.773	0.331	**0.019**	**0.021**	**0.005**
C-r&g	0.355	0.486	0.127	0.257	0.182	0.031	0.297	0.218	0.038	0.273	0.527	0.130	0.711	0.707	0.214	**0.035**	**0.047**	**0.009**
avg	0.389	0.534	0.124	0.268	0.202	0.025	0.239	0.160	0.028	0.413	0.598	0.167	0.612	0.634	0.192	**0.017**	**0.022**	**0.004**
B-a	0.679	0.795	0.118	0.407	0.345	0.023	0.093	0.072	0.011	0.738	0.653	0.055	0.896	0.869	0.145	**0.003**	**0.003**	**0.002**
L-g	0.417	0.330	0.022	0.246	0.138	0.013	0.258	0.176	0.014	0.379	0.239	0.014	0.622	0.512	0.055	**0.005**	**0.005**	**0.003**
L-r	0.728	0.730	0.062	0.494	0.331	0.040	0.187	0.130	0.023	0.651	0.441	0.029	0.816	0.801	0.056	**0.020**	**0.020**	**0.005**
L-g&r	0.844	0.822	0.091	0.606	0.441	0.035	0.184	0.112	0.011	0.597	0.454	0.032	0.904	0.888	0.191	**0.028**	**0.028**	**0.009**
avg	0.663	0.627	0.058	0.449	0.303	0.029	0.210	0.139	0.016	0.542	0.378	0.025	0.781	0.734	0.101	**0.018**	**0.018**	**0.006**
Cre-a	0.363	0.434	0.266	0.120	0.080	0.032	0.082	0.042	0.017	0.186	0.267	0.070	0.225	0.349	0.220	**0.019**	**0.019**	**0.009**
Cre-g	0.156	0.192	0.115	0.075	0.040	0.041	0.157	0.089	0.050	0.199	0.254	0.118	0.125	0.216	0.075	**0.003**	**0.003**	**0.031**
Cre-a&g	0.408	0.469	0.407	0.175	0.131	0.074	0.112	0.051	0.033	0.273	0.389	0.181	0.475	0.453	0.387	**0.059**	**0.059**	**0.002**
avg	0.309	0.365	0.263	0.123	0.084	0.049	0.117	0.061	0.033	0.219	0.303	0.123	0.275	0.339	0.227	**0.027**	**0.027**	**0.014**
Com-g	0.692	0.655	0.003	0.498	0.379	0.014	0.506	0.431	0.017	0.638	0.641	0.127	0.735	0.658	0.707	**0.010**	**0.010**	**0.002**
Com-r	0.554	0.542	0.075	0.476	0.465	0.002	0.467	0.486	0.006	0.339	0.494	0.067	0.480	0.554	0.011	**0.007**	**0.007**	**0.0**
Com-g&r	0.698	0.682	0.348	0.486	0.385	0.014	0.537	0.526	0.025	0.723	0.724	0.145	0.749	0.742	0.611	**0.109**	**0.109**	**0.011**
avg	0.142	0.648	0.626	0.487	0.410	0.010	0.503	0.481	0.016	0.567	0.620	0.113	0.655	0.651	0.443	**0.042**	**0.042**	**0.004**

4.5 RQ3: Deep Analysis on Repaired Models

Setup. This evaluation further studies why our method can outperform other methods. The decision process of DNN can be regarded as the information discarding process layer by layer. As described in Section 2.3, we will evaluate the layer-wise distance between the original data x and the new data x' that changes the protected attributes. Intuitively, we expect that the distance should be smaller, indicating that the protected attributes do not affect the prediction too much. Specifically, for each layer, we calculate the average L1 distance as follows:

$$D_l = \frac{\sum_{x \in X} \sum_{i \in I} \left| f_l(x) - f_l(x'_i) \right|_1}{|X| * |I|},$$

where l is the layer of the DNN f, I denotes the value combination set of the protected attributes, x'_i denotes the new data that changes values of the protected attributes to i in x, $|I|$ denotes the total number of possible value combinations of the protected attributes, and $|X|$ is the total number of the data X. We here show the results on the Census Income dataset, Bank Marketing dataset, and LSAC dataset.

Results. Table 5 shows the detailed results. Specifically, we calculate the layer distance except for the input layer and the output layer of the model. Except for our models, we also show the results of the vanilla models and the repaired models by M_{dis} for reference. M_{dis} is selected because it achieved better results in the baselines. The results show that, compared with vanilla models, our repaired models tend to reduce the average distance in most cases. However, we found that the average distance from M_{dis} is much smaller than that of the vanilla model and ours from layer 2 to layer 5. The reason could be that similar to adversarial retraining, M_{dis} introduces many discriminatory instances that can directly affect the feature distance by training. However, our method works on the neuron level that indirectly affects the feature distance in the hidden layers. Considering the last layer, we can clearly see that the average distance of *Faire* is much smaller than others. This phenomenon is consistent with the salient effectiveness of *Faire*, i.e., the smaller the average distance in the logit layer, i.e., the last layer, the fairer the repaired model. It also shows that, although our neuron-based analysis does not reduce the distance in the middle layers, it can largely reduce the distance in the last layer, which is more important for the final result. We conjecture that the neuron-based analysis is based on DeepLIFT, which performs the backward

Table 5. Difference between the Middle Layer Features of the Original Data and Flipped Data

Data	layer2			layer3			layer4			layer5			layer6		
	M_{van}	M_{dis}	*Faire*	M_{van}	M_{dis}	*Faire*	M_{van}	M_{dis}	*Faire*	M_{van}	M_{dis}	*Faire*	M_{van}	M_{dis}	*Faire*
C-a	0.296	0.157	**0.070**	0.138	0.069	**0.031**	0.273	0.055	**0.049**	0.170	0.017	**0.012**	0.138	0.021	**0.000**
C-r	0.237	**0.063**	0.236	0.152	**0.039**	0.146	0.233	**0.032**	0.169	0.150	**0.016**	0.119	0.106	0.011	**0.002**
C-g	0.135	0.055	**0.028**	0.091	**0.033**	0.044	0.169	**0.025**	0.038	0.112	**0.011**	0.025	0.082	0.008	**0.001**
C-a&r	0.393	**0.186**	0.213	0.169	**0.097**	0.210	0.145	**0.069**	0.153	0.093	**0.030**	0.035	0.141	0.029	**0.004**
C-a&g	0.368	0.169	**0.102**	0.154	0.076	**0.040**	0.240	0.061	**0.021**	0.149	0.028	**0.011**	0.138	0.025	**0.004**
C-r&g	0.289	**0.085**	0.239	0.175	**0.055**	0.188	0.162	**0.036**	0.226	0.099	**0.015**	0.143	0.112	0.016	**0.009**
B-a	0.433	**0.191**	0.226	0.260	**0.092**	0.206	0.246	**0.041**	0.082	0.240	**0.018**	0.023	0.120	0.029	**0.000**
L-g	**0.047**	0.111	0.068	0.077	0.037	**0.023**	0.121	0.027	**0.026**	0.084	0.026	**0.010**	0.089	0.017	**0.001**
L-r	**0.141**	0.177	0.145	0.220	**0.066**	0.067	0.317	0.050	**0.047**	0.198	0.050	**0.042**	0.178	0.036	**0.001**
L-g&r	**0.169**	0.220	0.249	0.231	**0.101**	0.166	0.356	**0.067**	0.093	0.180	0.064	**0.055**	0.165	0.047	**0.002**

analysis on the neuron contributions from the last layer. Therefore, *Faire* could have a large impact on the output of the last layer.

Answer to RQ3: The discriminatory instances retraining approach can generally reduce the average distance in each layer. Differently, without using discriminatory instances, *Faire* does not reduce the feature distance consistently in the middle layers but reduces the distance of the last layer a lot. The results of the last layer can directly affect fairness, which explains why our method is more effective than others.

4.6 RQ4: Efficiency of *Faire*

For discriminatory instance retraining methods, we divide the repair process into data generation (T_{data}) and training (T_{train}) to count the time overhead, respectively (e.g., M_{dis}, M_a). For M_{flip} and M_{mt}, we only take training (T_{train}) time into account. For *Faire*, we separately count the time spent in training the protected attribute classifier (T_{train_p}), neuron analysis ($T_{analysis}$), and final repairing (T_{repair}). Table 6 shows the detailed time overhead on the Census Income dataset, Bank Marketing dataset, and LSAC dataset. We observe that to generate all data needed to retrain a M_{dis} model, it costs even more than 3 days if we generate for all attribute combinations in sequence. For M_a, the time overhead could be more than 1 day (i.e., generation for a&r attributes combination). For M_{flip} and M_{mt}, only training time is involved and the training time is no more than 80 s. For *Faire*, training for protected attributes costs no more than 190 s. Meanwhile, it takes less than 120 s to calculate contribution scores via DeepLIFT. The retraining takes less than 50 s, and the total process takes less than 340 s. Even though our method needs to search the optimal parameters lb and ub, the searching could be easily done in parallel, which will reduce the introduced time overhead.

Answer to RQ4: Although flipping-based methods to retrain and multitask learning are very efficient, their effectiveness is inadequate. Compared with discriminatory instances retraining methods, which are relatively effective, *Faire* is far more efficient. The total time overhead demonstrates the efficiency of our method.

5 THREATS TO VALIDITY AND DISCUSSION

Threats. The selection of datasets and models could be a threat that our method may not generalize to other datasets well. To counter this issue, we selected three popular benchmarks and architectures that are used in state-of-the-art testing techniques. The usage of the interpretation tool DeepLIFT could also be a threat to our method. Randomness existing in the evaluation process could be a threat. We repeat the evaluation process many times to reduce the effect of randomness. The internal threat lies in our implementation, and we carefully checked our code.

Table 6. Comparison on Time Overhead (Seconds)

Dataset	Attr	M_{dis}			M_a			M_{flip}	M_{mt}	*Faire*			
		T_{data}	T_{train}	T_{total}	T_{data}	T_{train}	T_{total}	T_{train}	T_{train}	T_{train_p}	$T_{analysis}$	T_{repair}	T_{total}
Census	a	301,341.1	254.3	301,595.4	49,538.0	137.9	49,675.9	58.0	52.4	87.4	52.3	36.4	176.1
	g	301,341.1	254.3	301,595.4	10,724.1	132.2	10,856.3	51.2	52.8	94.3	53.7	32.5	180.5
	r	301,341.1	254.3	301,595.4	18,555.7	148.7	18,704.4	49.3	52.8	88.6	59.2	29.8	177.6
	a&g	301,341.1	254.3	301,595.4	44,060.5	168.5	44,229.0	59.8	54.2	181.7	114.6	42.3	338.6
	a&r	301,341.1	254.3	301,595.4	110,551.9	233.2	110,785.1	79.5	66.3	176	117.4	45.3	338.7
	r&g	301,341.1	254.3	301,595.4	67,910.9	172.5	68,083.4	62.2	56.5	182.9	109.8	41.2	333.9
Bank	a	94,645.6	126.3	94,771.9	94,645.6	126.2	94,771.8	55.9	51.2	80.7	51.0	36.9	168.6
LSAC	g	158,461.2	88.5	158,549.7	17,755.8	111.8	17,867.6	28.5	35.8	54.4	46.8	22.1	123.3
	r	158,461.2	88.5	158,549.7	56,930.7	125.3	57,056.0	30.7	33.9	50.0	47.2	24.2	121.4
	g&r	158,461.2	88.5	158,549.7	83,774.7	176.1	83,950.8	42.5	38.9	104.4	92.9	26.5	223.8

Discussion. Although *Faire* has achieved promising results in fairness repair, there are still some limitations that can be further enhanced:

- *The lower accuracy.* From the results in RQ1, we found that we reduce more accuracy than others. Our deep analysis reveals that this might be caused by the rough setting, i.e., we set the same lower bound and the upper bound for all neurons. In the future, we could extend our method to set suitable lower/upper bounds for different neurons.
- *Introduce additional data.* From the results in RQ2 and RQ3, we found that *Faire* did not well reduce the feature distance in the middle layers and still suffers from subtle individual discrimination. Previous methods using data augmentation (data-driven methods) could also improve fairness performance. Intuitively, we can combine our method with the data-driven method to further improve fairness. In the future, we will explore the combination of *Faire* with the data-driven method.
- *Support other neural networks.* In this article, we mainly focus on the fully connected neural network that is widely used in the existing individual fairness research. In the future, we try to extend our method to other networks (e.g., Recurrent Neural Networks on sequential datasets and Convolutional Neural Networks on image datasets).
- *Feature representation.* In this work, we use neuron behaviors to distinguish features and then reduce the features related to protected attributes. However, our approach is not limited to feature analysis at neuron granularity. We could explore other feature representation analysis methods of multi-granularity [32]. Such methods could potentially better separate protected features and unprotected features apart, which might enable us to better penalize the unprotected features and promote the protected features.
- *Support other RGB image datasets.* Conducting a neuron-granularity function analysis can help locate most of the responsible neurons. However, for certain delicate discrimination features like color, biases may exist in a large number of neurons that are challenging to identify and repair using an additional condition layer. This is because the processing of such features may be distributed across a complex network of neurons. As a result, pinpointing the exact neurons responsible for the bias can be difficult. Moreover, due to the intricacies of these features, attempting to adjust them without affecting other aspects of neural processing can be challenging.

6 RELATED WORK

6.1 Neuron-based Deep Learning Model Analysis

In our work, we analyze neurons' functionality based on existing explanation techniques. There is already a line of work analyzing middle representations (i.e., the middle layers outputs of deep learning models) at the neuron level. We here use "accountable neurons" to unify different aliases to neurons critical to model classification results. References [29, 56] focus on exploring neurons accountable to adversarial attacks (i.e., neurons critical to the wrong classification results under the adversarial attack). In Reference [29], they extract accountable neurons via layer-wise gradient calculation and they prove that these accountable neurons are both crucial to adversarial attack and generalization performance. In Reference [51], they utilize **layer-wise relevance propagation (LRP)** [7] to calculate neuron relevance to the final classification, and higher relevance means higher importance. They analyze the accountable neurons to construct the decision structure of a DNN. They then design new coverage criteria (i.e., metrics to evaluate the effectiveness of a test suite) based on the constructed decision structure. This article calculates neurons that could represent decision logic in DNNs. In Reference [22], they quantify the contribution of individual data points by a "zero-out" strategy (i.e., remove individual data points completely by setting data points as zero). They then identify neurons vulnerable to adversaries and discover unfair neurons, while the calculation is complex and time-consuming. In Reference [60], the authors slice deep neural networks based on data flow analysis and quantify the neuron contributions to the slicing criterion. Moreover, they calculate accountable neurons according to the neuron contributions, which also involves a huge time overhead. In Reference [47], the authors train a gate for each neuron (an associated control gate to each layer's neuron) to identify critical neurons, and this involves a retraining process.

In this article, we aim to locate accountable neurons for original tasks and protected attributes classification tasks efficiently at the same time. We here adopted DeepLIFT to help with our analysis for its reliability and convenience for deployment. We did a further analysis based on the roles neurons play in two tasks.

6.2 Fairness Testing and Repair

There is already a line of works [8, 18, 28, 36, 38, 39, 53, 55] focusing on testing and improving group fairness, where examples are grouped according to a particular sensitive attribute, and statistics are calculated across groups.

However, the evaluations of group fairness and individual fairness are totally different [10]. In our article, we focus on individual fairness. Reference [20] defines fairness and discrimination and develops THEMIS to generate efficient test suites to measure discrimination. To improve the inefficiency problem of THEMIS caused by the random sampling process, Reference [45] proposes AEQUITAS to split the generation process into global generation and local generation for better guiding the exploration. Global generation is to discover discriminatory instances by randomly exploring the input space, while local generation perturbs explored instances in the global phase by three well-designed strategies to generate more discriminatory instances. They then propose to retrain a fairer model with generated discriminatory instances. Reference [2] uses LIME [40] to guide the generation of discriminatory instances. Reference [59] involves adversarial methods in the global searching process. They utilize the characteristic that adversarial attacks can gradually generate instances near the decision boundary to generate discriminatory instances efficiently in the global phase. Reference [58] improves ADF both in the global phase and local phase. They boost global generation with momentum and improve local generation with **Effective Vicinity Explorer (EVE)** to make the exploration more effective, and thus improve both effectiveness and efficiency. Reference [49] generates natural individual discriminatory instances with the help of a generative adversarial network. In References [58, 59], authors integrate the generated discriminatory instances to retrain given models to improve fairness. However, this repairing method is far from efficient due to the huge time overhead in the generating process. Several methods have been proposed to repair DNNs. Reference [42] directly manipulates the neuron weights without retraining to repair. Reference [21] designs guided test generation techniques to do data augmentation to repair. Reference [54] mutates styles of images to achieve data augmentation to repair. However, all these works target robustness instead of fairness. In our article, we expect to repair the given models on fairness efficiently without generating extra data.

7 CONCLUSION

In this article, we propose *Faire* to repair individual discrimination in DNNs. Based on the empirical study to assess the effect of protected features on individual discrimination, we infer the feature-level condition that will filter the protected features to repair DNNs. Different from existing methods, *Faire* does not need any additional data. Instead, we perform the neuron-based analysis and add a patch (a well-designed layer) to reduce the impact of features relevant to protected attributes in the prediction. Extensive experiments demonstrated the effectiveness and efficiency of our method. Although effective and efficient, it is time-consuming to fully explore the parameter space (e.g., unlimited combinations of lb and ub). In the future, we will find a method to guide the parameter setting.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Retrieved from https://www.tensorflow.org/ Software available from tensorflow.org.

[2] Aniya Aggarwal, Pranay Lohia, Seema Nagar, Kuntal Dey, and Diptikalyan Saha. 2018. Automated test generation to detect individual discrimination in AI models. Retrieved from http://arxiv.org/abs/1809.03260

[3] Aniya Aggarwal, Pranay Lohia, Seema Nagar, Kuntal Dey, and Diptikalyan Saha. 2019. Black box fairness testing of machine learning models. In *Proceedings of the 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 625–635.

[4] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. 2016. Concrete problems in AI safety. Retrieved from https://arXiv:1606.06565

[5] Anonymous [n.d.]. FairRepair. Retrieved from https://sites.google.com/view/fair-repair--anonymous/home

[6] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. 2019. Invariant risk minimization. Retrieved from https://arXiv:1907.02893

[7] S. Bach, Alexander Binder, Grégoire Montavon, F. Klauschen, K. Müller, and W. Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE* 10 (2015).

[8] Hyojin Bahng, Sanghyuk Chun, Sangdoo Yun, Jaegul Choo, and Seong Joon Oh. 2019. Learning de-biased representations with biased representations. Retrieved from http://arxiv.org/abs/1910.02806

[9] Rachel K. E. Bellamy, Kuntal Dey, Michael Hind, Samuel C. Hoffman, Stephanie Houde, Kalapriya Kannan, Pranay Lohia, Jacquelyn Martino, Sameep Mehta, Aleksandra Mojsilovic, Seema Nagar, Karthikeyan Natesan Ramamurthy, John Richards, Diptikalyan Saha, Prasanna Sattigeri, Moninder Singh, Kush R. Varshney, and Yunfeng Zhang. 2018. AI Fairness 360: An Extensible Toolkit for Detecting, Understanding, and Mitigating Unwanted Algorithmic Bias. Retrieved from https://arxiv.org/abs/1810.01943

[10] Reuben Binns. 2019. On the apparent conflict between individual and group fairness. Retrieved from http://arxiv.org/abs/1912.06883

[11] Francois Chollet et al. 2015. *Keras*. Retrieved from https://github.com/fchollet/keras

[12] Datamonsters. 2017. 10 Applications of Artificial Neural Networks in Natural Language Processing. Retrieved from https://medium.com/@datamonsters/artificial-neural-networks-in-natural-language-processing-bcf62aa9151a

[13] Mengnan Du, Fan Yang, Na Zou, and Xia Hu. 2019. Fairness in deep learning: A computational perspective. Retrieved from http://arxiv.org/abs/1908.08843

[14] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. Retrieved from http://archive.ics.uci.edu/ml

[15] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard S. Zemel. 2011. Fairness through awareness. Retrieved from http://arxiv.org/abs/1104.3913

[16] Carnegie Endowment. 2019. The Global Expansion of AI Surveillance. Retrieved from https://carnegieendowment.org/2019/09/17/global-expansion-of-ai-surveillance-pub-79847

[17] Michael Feldman, Sorelle Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. 2015. Certifying and removing disparate impact. Retrieved from https://1412.3756

[18] Rui Feng, Yang Yang, Yuehan Lyu, Chenhao Tan, Yizhou Sun, and Chunping Wang. 2019. Learning fair representations via an adversarial framework. Retrieved from http://arxiv.org/abs/1904.13341

[19] 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*. 498–510.

[20] Sainyam Galhotra, Yuriy Brun, and Alexandra Meliou. 2017. Fairness testing: Testing software for discrimination. Retrieved from http://arxiv.org/abs/1709.03221.

[21] Xiang Gao, Ripon K. Saha, Mukul R. Prasad, and Abhik Roychoudhury. 2020. Fuzz testing-based data augmentation to improve robustness of deep neural networks. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering (ICSE'20)*. ACM, New York, NY, 1147–1158. https://doi.org/10.1145/3377811.3380415

[22] Amirata Ghorbani and James Y. Zou. 2020. Neuron shapley: Discovering the responsible neurons. Retrieved from https://arxiv.org/abs/2002.09815

[23] Ben Goldberger, Guy Katz, Yossi Adi, and Joseph Keshet. 2020. Minimal modifications of deep neural networks using verification. In *Proceedings of the 23rd International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'20)*.

[24] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. Retrieved from http://www.deeplearningbook.org

[25] Dan Hendrycks and Thomas Dietterich. 2019. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*.

[26] Preethi Lahoti, Alex Beutel, Jilin Chen, Kang Lee, Flavien Prost, Nithum Thain, Xuezhi Wang, and Ed H. Chi. 2020. Fairness without demographics through adversarially reweighted learning. Retrieved from https://arXiv:2006.13114

[27] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324. https://doi.org/10.1109/5.726791

[28] Tianlin Li, Qing Guo, Aishan Liu, Mengnan Du, Zhiming Li, and Yang Liu. 2023. FAIRER: Fairness as Decision Rationale Alignment. Retrieved from https://2306.15299

[29] Tianlin Li, Aishan Liu, Xianglong Liu, Yitao Xu, Chongzhi Zhang, and Xiaofei Xie. 2020. Understanding adversarial robustness via critical attacking route. *Info. Sci.* 547 (Aug. 2020). https://doi.org/10.1016/j.ins.2020.08.043

[30] Suyun Liu and Luís Nunes Vicente. 2020. Accuracy and fairness trade-offs in machine learning: A stochastic multi-objective approach. Retrieved from https://arxiv.org/abs/2008.01132

[31] Haotian Ma, Yinqing Zhang, Fan Zhou, and Quanshi Zhang. 2019. Quantifying layerwise information discarding of neural networks. Retrieved from http://arxiv.org/abs/1906.04109

[32] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang Liu et al. 2018. Deepgauge: Multi-granularity testing criteria for deep learning systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. 120–131.

[33] Tom Van Mele et al. 2017–2021. COMPAS: A framework for computational research in architecture and structures. Retrieved from http://compas.dev https://doi.org/10.5281/zenodo.2594510

[34] BBC News. 2020. *IBM Abandons "Biased" Facial Recognition Tech.* Retrieved from https://www.bbc.co.uk/news/technology-52978191

[35] BBC News. 2021. *AI at Work: Staff "Hired and Fired by Algorithm."* Retrieved from https://www.bbc.com/news/technology-56515827

[36] Luca Oneto, Michele Donini, Amon Elders, and Massimiliano Pontil. 2020. Taking Advantage of Multitask Learning for Fair Classification. Retrieved from https://1810.08683

[37] Hua Qi, Zhijie Wang, Qing Guo, Jianlang Chen, Felix Juefei-Xu, Lei Ma, and Jianjun Zhao. 2021. ArchRepair: Block-Level Architecture-Oriented Repairing for Deep Neural Networks. Retrieved from https://2111.13330

[38] Vikram V. Ramaswamy, Sunnie S. Y. Kim, and Olga Russakovsky. 2021. Fair attribute classification through latent space de-biasing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'21)*.

[39] Mhd Hasan Sarhan, Nassir Navab, Abouzar Eslami, and Shadi Albarqouni. 2020. Fairness by learning orthogonal disentangled representations. Retrieved from https://arxiv.org/abs/2003.05707

[40] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. FaceNet: A unified embedding for face recognition and clustering. Retrieved from http://arxiv.org/abs/1503.03832

[41] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning important features through propagating activation differences. Retrieved from http://arxiv.org/abs/1704.02685

[42] Jeongju Sohn, Sungmin Kang, and Shin Yoo. 2019. Search-based repair of deep neural networks. Retrieved from http://arxiv.org/abs/1912.12463

[43] Matthew Sotoudeh and Aditya V. Thakur. 2021. Provable repair of deep neural networks. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*. 588–603.

[44] TechCrunch. 2020. Nearly 70% of U.S. smart speaker owners use Amazon Echo devices. Retrieved from https://techcrunch.com/2020/02/10/nearly-70-of-u-s-smart-speaker-owners-use-amazon-echo-devices/

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

[46] Tianlu Wang, Jieyu Zhao, Mark Yatskar, Kai-Wei Chang, and Vicente Ordonez. 2019. Balanced datasets are not enough: Estimating and mitigating gender bias in deep image representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV'19)*.

[47] Yulong Wang, Hang Su, Bo Zhang, and Xiaolin Hu. 2018. Interpret neural networks by identifying critical data routing paths. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8906–8914. https://doi.org/10.1109/CVPR.2018.00928

[48] Z. Wang, K. Qinami, I. Karakozis, K. Genova, P. Nair, K. Hata, and O. Russakovsky. 2020. Towards fairness in visual recognition: Effective strategies for bias mitigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'20)*. IEEE, Los Alamitos, CA, 8916–8925. https://doi.org/10.1109/CVPR42600.2020.00894

[49] Yisong Xiao, Aishan Liu, Tianlin Li, and Xianglong Liu. 2023. Latent Imitator: Generating Natural Individual Discriminatory Instances for Black-Box Fairness Testing. Retrieved from https://2305.11602

[50] Xiaofei Xie, Wenbo Guo, Lei Ma, Wei Le, Jian Wang, Lingjun Zhou, Yang Liu, and Xinyu Xing. 2021. RNNrepair: Automatic RNN repair via model-based analysis. In *Proceedings of the International Conference on Machine Learning (ICML'21)*. PMLR, 11383–11392.

[51] Xiaofei Xie, Tianlin Li, Jian Wang, Lei Ma, Qing Guo, Felix Juefei-Xu, and Yang Liu. 2022. NPC: Neuron Path Coverage via characterizing decision logic of deep neural networks. *ACM Trans. Software Eng. Methodol.* 31, 3 (2022), 1–27.

[52] Yingfei Xiong, Jie Wang, Runfa Yan, Jiachen Zhang, Shi Han, Gang Huang, and Lu Zhang. 2017. Precise condition synthesis for program repair. In *Proceedings of the IEEE/ACM 39th International Conference on Software Engineering (ICSE'17)*. IEEE, 416–426.

[53] Tian Xu, Jennifer White, Sinan Kalkan, and Hatice Gunes. 2020. Investigating bias and fairness in facial expression recognition. Retrieved from https://arxiv.org/abs/2007.10075

[54] Bing Yu, Hua Qi, Qing Guo, Felix Juefei-Xu, Xiaofei Xie, Lei Ma, and Jianjun Zhao. 2020. DeepRepair: Style-guided repairing for DNNs in the real-world operational environment. Retrieved from https://arxiv.org/abs/2011.09884

[55] Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. 2018. Mitigating unwanted biases with adversarial learning. Retrieved from http://arxiv.org/abs/1801.07593

[56] Chongzhi Zhang, Aishan Liu, Xianglong Liu, Yitao Xu, Hang Yu, Yuqing Ma, and Tianlin Li. 2020. Interpreting and improving adversarial robustness of deep neural networks with neuron sensitivity. *IEEE Trans. Image Process.* 30 (2020), 1291–1304.

[57] Hao Zhang and W. K. Chan. 2019. Apricot: A weight-adaptation approach to fixing deep learning models. In *Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering (ASE'19)*. IEEE, 376–387.

[58] Lingfeng Zhang, Yueling Zhang, and Min Zhang. 2021. Efficient white-box fairness testing through gradient search. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA'21)*. ACM, New York, NY, 103–114. https://doi.org/10.1145/3460319.3464820

[59] 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 ACM/IEEE 42nd International Conference on Software Engineering (ICSE'20)*. ACM, New York, NY, 949–960. https://doi.org/10.1145/3377811.3380331

[60] Ziqi Zhang, Yuanchun Li, Yao Guo, Xiangqun Chen, and Yunxin Liu. 2020. Dynamic slicing for deep neural networks. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE'20)*. ACM, New York, NY, 838–850. https://doi.org/10.1145/3368089.3409676

Received 8 February 2022; revised 26 April 2023; accepted 12 June 2023