NeuronFair: Interpretable White-Box Fairness Testing through Biased Neuron Identification

Haibin Zheng  
Zhejiang University of Technology  
haibinzheng320@gmail.com

Zhiqing Chen  
Zhejiang University of Technology  
201706060201@zjut.edu.cn

Tianyu Du  
Zhejiang University  
zjradty@zjut.edu.cn

Xuhong Zhang  
Zhejiang University  
zhangxuhong@zju.edu.cn

Yao Cheng  
Huawei International Pte. Ltd.  
chengyao101@huawei.com

Shouling Ji  
Zhejiang University  
sji@zju.edu.cn

Jinyi Wang  
Zhejiang University  
wangjinee@zju.edu.cn

Yue Yu  
National University of Defense Technology  
yuyue@nudt.edu.cn

Jinyin Chen∗  
Zhejiang University of Technology  
chenjinyin@zjut.edu.cn

ABSTRACT
Deep neural networks (DNNs) have demonstrated their outperformance in various domains. However, it raises a social concern whether DNNs can produce reliable and fair decisions especially when they are applied to sensitive domains involving valuable resource allocation, such as education, loan, and employment. It is crucial to conduct fairness testing before DNNs are reliably deployed to such sensitive domains, i.e., generating as many instances as possible to uncover fairness violations. However, the existing testing methods are still limited from three aspects: interpretability, performance, and generalizability. To overcome the challenges, we propose NeuronFair, a new DNN fairness testing framework that differs from previous work in several key aspects: (1) interpretable - it quantitatively interprets DNNs’ fairness violations for the biased decision; (2) effective - it uses the interpretation results to guide the generation of more diverse instances in less time; (3) generic - it can handle both structured and unstructured data. Extensive evaluations across 7 datasets and the corresponding DNNs demonstrate NeuronFair’s superior performance. For instance, on structured datasets, it generates much more instances (~×5.84) and saves more time (with an average speedup of 534.56%) compared with the state-of-the-art methods. Besides, the instances of NeuronFair can also be leveraged to improve the fairness of the biased DNNs, which helps build more fair and trustworthy deep learning systems. The code of NeuronFair is open-sourced at https://github.com/haibinzheng/NeuronFair.

CCS CONCEPTS
• Computing methodologies → Artificial intelligence; • Software and its engineering → Software reliability.

KEYWORDS
Interpretability, fairness testing, discriminatory instance, deep learning, biased neuron

ACM Reference Format:
Haibin Zheng, Zhiqing Chen, Tianyu Du, Xuhong Zhang, Yao Cheng, Shouling Ji, Jingyi Wang, Yue Yu, and Jinyin Chen. 2021. NeuronFair: Interpretable White-Box Fairness Testing through Biased Neuron Identification. In Proceedings of The 44th International Conference on Software Engineering (ICSE 2022). ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/nnnnnnnn.nnnnnnn

1 INTRODUCTION
Deep neural networks (DNNs) [38] have been increasingly adopted in many fields, including computer vision [5], natural language processing [19], software engineering [13, 18, 32, 39, 45, 48], etc. However, one of the crucial factors hindering DNNs from further serving applications with social impact is the unintended individual discrimination [44, 47, 55]. Individual discrimination exists when a given instance different from another only in sensitive attributes (e.g., gender, race, etc.) but receives a different prediction outcome from a given DNN [3]. Taking gender discrimination in salary prediction as an example, for two identical instances except for the gender attribute, male’s annual income predicted by the DNN is often higher than female’s [35]. Thus, it is of great importance for stakeholders to uncover fairness violations and then to reduce DNNs’ discrimination so as to responsibly deploy fair and trustworthy deep learning systems in many sensitive scenarios [12, 25, 30, 42].

Much effort has been put into uncovering fairness violations [7, 8, 11, 16, 23, 26, 29, 43, 53]. The most common method is fairness testing [2, 3, 9, 22, 24, 52, 56, 57], which solves this problem by generating as many instances as possible. Initially, fairness testing is designed to uncover and reduce the discrimination in traditional machine learning (ML) with low-dimensional linear models. However, such methods are suffering from several problems. First, most of them (e.g., FairAware [22], BlackPT [3], and FlipTest [9]) cannot handle DNNs with high-dimensional nonlinear structures. Then, though some of them (e.g., Themis [24], SymbGen [2], and Aequitas [52]) can be applied to test DNNs, they are still challenged by the high time cost and numerous duplicate instances. Recently, several methods have been specifically developed for DNNs, such as ADF [57] and EIDIG [56], etc. These methods make progress in...
effectiveness and efficiency through gradient guidance, but they still suffer from the following problems.

First, these methods can hardly be generalized to unstructured data. As we know, DNNs are originally designed to process unstructured data (e.g., image, text, speech, etc.), but almost no existing fairness testing method can be applied to these data. It is mainly because these methods cannot determine which features are related to sensitive attributes, and cannot implement appropriate modifications to these features, e.g., how to determine pixels related to gender attribute in face images, and how to modify these pixel values to change gender [54]. However, even a seemingly simple task such as face detection [34] is subject to extreme amounts of fairness violations. It is especially concerning since these facial systems are often not deployed in isolation but rather as part of the surveillance or criminal detection pipeline [4]. Therefore, these testing methods still cannot serve DNNs widely until we solve the problem of data generalization.

Second, the generation effectiveness of these methods is challenged by gradient vanishing. They leverage the gradient-guided strategy to improve generation efficiency, but the gradient may vanish and cause instance generation to fail. Additionally, when the gradient is small, the generated instances are highly similar. However, the purpose of fairness testing is to generate not only the numerous instances, but also the diverse instances.

Third, almost all existing methods hardly provide interpretability. They only focus on generating numerous instances, but cannot interpret how the biased decisions occurred. DNNs’ decision results are determined by neuron activation, then we try to study these neurons that cause biased decisions. We find that the instances generated by existing testing methods will miss the coverage of these neurons that cause biased decisions (refer to the experiment result in Fig. 6). More seriously, we cannot even know which neurons related to biased decisions have been missed for testing when there is a lack of interpretability. Therefore, we need an interpretable testing method so as to interpret DNNs’ biased decisions and evaluate instances’ utility for uncovering fairness violations. Based on these, the interpretation results can guide us to design effective testing to uncover more discrimination. In summary, the current fairness testing challenges lie in the lack of data generalization, generation effectiveness, and discrimination interpretation.

To overcome the above challenges, our design goals are as follows: 1) we intend to uncover and quantitatively interpret DNNs’ discrimination; 2) then, we plan to apply this interpretation results to guide fairness testing; 3) furthermore, we want to generalize our testing method to unstructured data. Due to the decision results of DNNs are determined by the nonlinear combination of each neuron’s activation state, thus we imagine whether the biased decisions are caused by some neurons. Then, we try to observe the neuron activation state in DNNs’ hidden layers through feeding instance pair, which is two identical instances except for the sensitive attribute. Surprisingly, we find that the activation state follows such a pattern, i.e., neurons with drastically varying activation values are overlapping for different instance pairs. We observe that DNNs’ discrimination is reduced when these overlapped neurons are zeroed out. Therefore, we conclude that these neurons cause the DNNs’ discrimination. Then, we intend to quantitatively interpret DNNs’ discrimination by computing the neuron activation difference (ActDiff) values.

According to the interpretation results, we further design a testing method, NeuronFair, to optimize gradient guidance. First, we determine the main neurons that cause discrimination, called biased neurons. Then, we search for discriminatory instances with the optimization object of increasing the ActDiff values of biased neurons. Because the optimization from the biased neuron shortens the derivation path, it reduces the probability of the gradient vanishing and time cost. Moreover, we can produce more diverse instances through the dynamic combination of biased neurons. All in all, we leverage the interpretation results to optimize gradient guidance, which is beneficial to the generation effectiveness.

We leverage adversarial attacks [14, 27, 36] to determine which features are related to sensitive attributes, and make appropriate modifications to these features. The adversarial attack is originally to test the DNNs’ security, e.g., slight modifications to some image pixels will cause the predicted label to flip [10, 15, 20, 50]. Taking the gender attribute of face image as an example, we consider training a classifier with ‘male’ and ‘female’ as labels, then adding the perturbation to the face image until its predicted gender label flips. Based on this generalization framework, we can modify the sensitive attributes of any data, thereby generalizing NeuronFair to any data type.

In summary, we first implement to quantitatively interpret the discrimination using neuron-based analysis; then, we leverage the interpretation results to optimize the instance generation; finally, we design a generalization framework for sensitive attribute modification. The main contributions are as follows.

- Through the neuron activation analysis, we quantitatively interpret DNNs’ discrimination, which provides a new perspective for measuring discrimination and guides DNNs’ fairness testing.
- Based on the interpretation results, we design a novel method for DNNs’ discriminatory instance generation, NeuronFair, which significantly outperforms previous works in terms of effectiveness.
- Inspired by adversarial attacks, we design a generalization framework to modify sensitive attributes of unstructured data, which generalizes NeuronFair to unstructured data.
- We publish our NeuronFair as a self-contained open-source toolkit online.

2 BACKGROUND

To better understand the problem we are tackling and the methodology we propose in later sections, we first introduce DNN, data form, individual discrimination, and our problem definition.

**DNN.** A DNN can be represented as \( f(x; \Theta) : X \rightarrow Y \), including an input layer, several hidden layers, and an output layer. Two popular architectures of DNNs are fully connected network (FCN) and convolutional neural network (CNN). For a FCN, we denote the activation output of each neuron in the hidden layer as: \( f^k_l(x; \Theta) \), where \( \Theta \) is the weights, \( l \in \{1, 2, ..., N_l\} \), \( N_l \) is the number of neural layers, \( k \in \{1, 2, ..., N_k\} \), \( N_k \) is the number of neurons in the \( l \)-th layer. For a CNN, we flatten the output of the convolutional layer for the calculation of neuron activation. The loss function of DNNs
with respect to attribute $A$, where
\begin{equation}
\Delta \text{senatt} = \text{the perturbation added to the gender attribute to flip gender. (ii) Discrimination exists when the instance’s predicted label changes as the gender attribute is flipped, i.e., the instance crosses the decision boundary.}
\end{equation}

is defined as follows:
\begin{equation}
f(x, y; \Theta) = -\frac{1}{N} \left[ \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} (y_{i,j} \times \log(\hat{y}_{i,j})) \right]
\end{equation}

where $N$ is the number of instances, $M$ is the number of classes, $y_i$ is the ground-truth of $x_i$, $\hat{y}_i = f(x_i; \Theta)$ is the predicted probability, $\log(\cdot)$ is a logarithmic function.

Data Form. Denote $X = \{x_i\}$, $Y = \{y_i\}$ as a normal dataset, and its instance pairs by $<X, X'> = \{<x_i, x'_i>\}, i \in \{0, 1, ..., N-1\}$. For an instance, we denote its attributes by $A = \{a_i\}, i \in \{0, 1, ..., N-1\}$, where $A_s \subset A$ is a set of sensitive attributes, and $A_{ns} = \{a_i^{ns} | a_i^{ns} \in A, a_i^{ns} \notin A_s\}$ is a set of non-sensitive attributes. Note that sensitive attributes (e.g., gender, race, age, etc.) are usually given in advance according to specific sensitive scenes.

Individual Discrimination. As stated in previous work [3, 22, 37], individual discrimination exists when two valid inputs which differ only in the sensitive attributes but receive a different prediction result from a given DNN, as shown in Fig. 1. Such two valid inputs are called individual discriminatory instances (IDIs).

Definition 1: IDI determination. We denote $<X_d, X'_d> = \{<x_{d,i}, x'_{d,i}>\}$ as a set of IDI pairs, which satisfies:
\begin{equation}
f(x_{d,i}; \Theta) \neq f(x'_{d,i}; \Theta)
\end{equation}
s.t. $x_{d,i}[A_s] \neq x'_{d,i}[A_s], x_{d,i}[A_{ns}] = x'_{d,i}[A_{ns}]
\end{equation}

where $i \in \{0, 1, ..., N_d - 1\}, x_{d,i}[A_s]$ represents the value of $x_{d,i}$ with respect to attribute $A_s$. Note that our instances are generative (e.g., maybe the age of a generated instance is 150 years old on Adult dataset), thus we need to clip their attribute values that do not exist in the input domain $I$.

Problem Definition. A DNN which suffers from individual discrimination may produce biased decisions when an IDI is presented as input. Below are the three goals that we want to achieve through the devised fairness testing technique. First, observe the activation state of neurons, find the correlation between neuron activation pattern and biased decision, and interpret the reason for discrimination. Then, based on the interpretation results, we generate IDIs with maximizing DNN’s discrimination as the optimization object. In the generation process, we consider not only the generation quantity, but also the diversity. Finally, we conduct fairness testing on unstructured datasets.

3 NEURONFAIR

An overview of NeuronFair is presented in Fig. 2. NeuronFair has two parts, i.e., discrimination interpretation and IDI generation based on interpretation results. During the discrimination interpretation, we first interpret why discrimination exists through neuron-based analysis. Then, we design a discrimination metric based on the interpretation result, i.e., AUC value, as shown in Fig. 2 (i). AUC is the area under AS curve, where the AS curve records the percentage of neurons above the ActDiff threshold. Finally, we leverage the AS curve to adaptively identify biased neurons, which serves for IDI generation. During the IDI generation, we employ the biased neurons to perform global and local generations. The global phase guarantees the diversity of the generated instances, and the local phase guarantees the quantity, as shown in Fig. 2 (ii). On the one hand, the global generation uses the normal instance as a seed and stops if an IDI is generated or it times out. On the other hand, the generated IDIs are adopted as seeds of local generation, leading to generate as many IDIs as possible near the seeds. Besides, we implement dynamic combinations of biased neurons to increase diversity, and use the momentum strategy to accelerate IDI generation. In the following, we first quantitatively interpret DNNs’ discrimination, then present details of IDI generation based on interpretation results, and finally generalize NeuronFair to unstructured data.

3.1 Quantitative Discrimination Interpretation

First, we draw AS curve and compute AUC value to measure DNNs’ discrimination. Then, based on the measurement results, we identify the key neurons that cause unfair decisions as biased neurons.

3.1.1 Discrimination Measurement. The ActDiff is calculated as follows:
\begin{equation}
z_k^l = \frac{1}{N} \sum_{i=0}^{N_d-1} \left[ \text{abs}(f^k_l(x_i; \Theta) - f^k_l(x'_i; \Theta)) \right]
\end{equation}
where $z_k^l$ is the ActDiff of the $k$-th neuron in the $l$-th layer, $l \in \{1, 2, ..., N_l\}$, $N_l$ is the layer number, $N$ is the number of normal instance pairs $<X, X'> = \{<x_i, x'_i>, i \in \{0, 1, 2, ..., N-1\} \}$, $\text{abs}(\cdot)$
returns an absolute value, $f_i^k(x; \Theta)$ returns the activation output of the $k$-th neuron in the $l$-th layer, $\Theta$ represents the model weights.

Based on Eq. (3), we plot AS curve and compute AUC value. We first compute each neuron’s ActDiff and normalize it by hyperbolic tangent function $\text{Tanh}$, as shown in Fig.3 (i). “L1” means the 1-st hidden layer of a DNN with 64 neurons. Then, we set several ActDiff thresholds at equal intervals, count the neuron percentages above the ActDiff thresholds, and then compute the area under AS curve as AUC value, as shown in Fig.3 (ii), where the $x$-axis is the ActDiff value normalized by $\text{Tanh}$ function, the $y$-axis is SenNeuR.

More specific operations on AS curve drawing and AUC calculation are shown in Algorithm 1. First, we compute the average ActDiff values of each neuron $z_i^l$ at line 1. In the loop from lines 7 to 9, for each neural layer, we get SenNeuR for plotting the AS curve. Then, we compute AUC value by integration at line 11.

Algorithm 1: AS curve drawing and AUC calculation.

1. Calculate the average ActDiff of each neuron: $z_i^l = \frac{1}{N} \sum_{n=1}^{N} \text{abs}(f_{i}^{l}(x_{1}; \Theta) - f_{i}^{l}(x_{1}'; \Theta))$
2. For $l = 1 : N_l$
3. $z_l = \text{Tanh}(z_l)$
4. max-$z$ = max($z_l$)
5. $x_{\text{tmp}} = 0 : \text{step-interval} : \text{max}-z$
6. $y_{\text{tmp}} = [ ]$
7. For count = 1 : length($x_{\text{tmp}}$)
8. $y_{\text{tmp}} = [y_{\text{tmp}}, \text{length}($find($z_l > x_{\text{tmp}}[\text{count}])$)]$
9. \text{End For}
10. $y_{\text{tmp}} = y_{\text{tmp}}/\text{length}($number of neurons in the most biased layer, the biased neurons satisfy the condition

$\frac{\text{ActDiff}}{\text{Tanh}}$ of each iteration of iterations for each seed $\frac{\text{actDiff}_{\max}}{\text{Tanh}}$ of each seed $\frac{\text{actDiff}_{\max}}{\text{Tanh}}$

$\frac{\text{actDiff}_{\max}}{\text{Tanh}}$ of each seed $\frac{\text{actDiff}_{\max}}{\text{Tanh}}$

$\frac{\text{actDiff}_{\max}}{\text{Tanh}}$ of each seed $\frac{\text{actDiff}_{\max}}{\text{Tanh}}$

$\frac{\text{actDiff}_{\max}}{\text{Tanh}}$ of each seed $\frac{\text{actDiff}_{\max}}{\text{Tanh}}$

$\frac{\text{actDiff}_{\max}}{\text{Tanh}}$ of each seed $\frac{\text{actDiff}_{\max}}{\text{Tanh}}$

$\frac{\text{actDiff}_{\max}}{\text{Tanh}}$ of each seed $\frac{\text{actDiff}_{\max}}{\text{Tanh}}$

$\frac{\text{actDiff}_{\max}}{\text{Tanh}}$ of each seed $\frac{\text{actDiff}_{\max}}{\text{Tanh}}$

$\frac{\text{actDiff}_{\max}}{\text{Tanh}}$ of each seed $\frac{\text{actDiff}_{\max}}{\text{Tanh}}$

$\frac{\text{actDiff}_{\max}}{\text{Tanh}}$ of each seed $\frac{\text{actDiff}_{\max}}{\text{Tanh}}$

$\frac{\text{actDiff}_{\max}}{\text{Tanh}}$ of each seed $\frac{\text{actDiff}_{\max}}{\text{Tanh}}$

$\frac{\text{actDiff}_{\max}}{\text{Tanh}}$ of each seed $\frac{\text{actDiff}_{\max}}{\text{Tanh}}$

$\frac{\text{actDiff}_{\max}}{\text{Tanh}}$ of each seed $\frac{\text{actDiff}_{\max}}{\text{Tanh}}$

$\frac{\text{actDiff}_{\max}}{\text{Tanh}}$ of each seed $\frac{\text{actDiff}_{\max}}{\text{Tanh}}$

$\frac{\text{actDiff}_{\max}}{\text{Tanh}}$ of each seed $\frac{\text{actDiff}_{\max}}{\text{Tanh}}$

$\frac{\text{actDiff}_{\max}}{\text{Tanh}}$ of each seed $\frac{\text{actDiff}_{\max}}{\text{Tanh}}$

$\frac{\text{actDiff}_{\max}}{\text{Tanh}}$ of each seed $\frac{\text{actDiff}_{\max}}{\text{Tanh}}$

$\frac{\text{actDiff}_{\max}}{\text{Tanh}}$ of each seed $\frac{\text{actDiff}_{\max}}{\text{Tanh}}$

$\frac{\text{actDiff}_{\max}}{\text{Tanh}}$ of each seed $\frac{\text{actDiff}_{\max}}{\text{Tanh}}$...
Algorithm 3: Local generation guided by biased neurons.

Input: IDI pairs \( S_N = \{<x_{d_l}, y_{d_l}^l, x_{d_l}^l>, ... > \}, l = \{0, 1, ..., N_g - 1\} \), initial set \( \Omega_I = \emptyset \), the maximum number of iterations for each seed \( \text{max}_\text{iter}_l \), the perturbation size of each iteration \( \text{step}_\text{size}_l \), the decay rate of momentum \( \mu_l \), step size for random disturbance \( r \), \( \text{step}_l \),

Output: A set of IDI pairs found locally \( \Omega_I \).

1. For \( i = 0 : N_g - 1 \)
2. Select seed \( <x', x'> \) from \( \Omega_I \), \( g_0 = g'_0 = 0 \).
3. For \( t = 0 : \text{max}_\text{iter}_l \)
   4. If \( (\text{mod}(\text{step}_\text{size}_l, \text{step}_l) == 0) \) Then
     5. \( r = \text{Rand}_{(0,1)}(Pr) \)
     6. End If
   7. \( g_{t+1} = \mu_l \times g_t + \nabla_x J_{dl}(x; \Theta) \)
   8. \( g'_t = \mu_l \times g'_t + \nabla'_x J_{dl}(x' ; \Theta) \)
   9. \( \text{dir} = \text{sign}(g_{t+1} + g'_{t+1}) \)
   10. \( P_{\text{dir}} = \text{SoftMax}([g_{t+1} + g'_{t+1}]^{-1}) \)
   11. For \( a^k \in A_N \)
     12. Generate a random number \( P_{\text{tmp}} \in (0,1] \).
     13. If \( (P_{\text{tmp}} < P_{\text{dir}}[a^k]) \) Then
       14. \( x[a^k] = x[a^k] + \text{dir}[a^k] \times \text{step}_\text{size}_l \)
     15. End If
   16. End For
   17. \( x = \text{Clip} (x, \Gamma) \)
   18. Create \( <x', x'> \) s.t. \( x[\text{As}] = x'[\text{As}] \).
   19. If \( (f(x; \Theta) \neq f(x'; \Theta)) \) Then
     20. \( \Omega_I = \Omega_I \cup <x', x'> \)
     21. End If
   22. End For
23. End For

output of the \( k \)-th neuron. \( p \) is the position of biased neurons, \( r \) is the position of randomly selected neurons to increase the dynamics of \( f_{dl}(x; \Theta) \), \( r = \text{Rand}_{(0,1)}(Pr) \), where \( \text{Rand}_{(0,1)}(Pr) \) returns a random vector with only ‘0’ or ‘1’. \( r \) has the same size as \( p \) and satisfies \( \sum \text{r} = \text{INT}(N^k \times p) \), where \( \text{INT}() \) returns an integer. Here, we set \( Pr = 5\% \). ‘\|’ means ‘or’, \( p_k = r_k = 0 \) if and only \( p_k = 0 \) and \( r_k = 0 \). The optimization object of IDI generation is: \( \text{arg max} \ J_{dl}(x; \Theta) \).

Algorithm 2 shows the details of global generation with momentum acceleration. We first adopt k-maps clustering function \( \text{KMeans}(X, N_k) \) to process \( X \) into \( N_k \) clusters, and then get seeds from clusters in a round-robin fashion at line 3. We update random vector \( r \) at equal intervals from lines 5 to 7, not only to increase the dynamics but also to avoid excessively disturbing the generation task. According to Definition 1, we determine the IDIs from lines 8 to 12. We employ the momentum acceleration operation at lines 13 and 14, which can effectivly use historical gradient and reduce invalid searches. Note that we keep the value of the sensitive attribute in \( x \) at line 16. Finally, we clip the value of \( x \) to satisfy the input domain I.

3.2.2 Local Generation. Since the local generation aims to find as many IDIs as possible near the seed, we increase the iteration number of each seed \( \text{max}_\text{iter}_l \), and reduce the bias perturbation added in each iteration, as shown in Algorithm 3. Compared to the global phase, the major difference is the loop from lines 11 to 16, where we add perturbation to the non-sensitive attributes of large gradients with a small probability. We automatically get the probability of adding perturbation to each attribute in \( x \) at line 10.

3.3 Generalization Framework on Unstructured Data

We intend to solve the challenge of \( A_s \) modification to generalize NeuronFair to unstructured data. Here we take image data as an example. Attributes of an image are determined by pixels with normalized values between 0 and 1, i.e., the input domain of images is \( I \). Motivated by the adversarial attack, we design a generalization framework to implement the image’s \( A_s \) modification, which modifies \( A_s \) through adding a small perturbation to most pixels, as shown in Fig. 4.

We consider a fairness testing scenario for face detection, which determines whether the input image contains a face. The face detector consists of a CNN module (i.e., Fig. 4 (i)) and a FCN module (i.e., Fig. 4 (ii)). As shown in Fig. 4, for a given face image \( x \) and a detector \( f(x; \Theta) \), there are three steps: (1) build a sensitive attribute classifier; (2) produce \( \Delta_{\text{senatt}} \) based on Eq. (5); \( \Delta_{\text{senatt}} \) is the perturbation added to image to flip sensitive attribute; (3) generate \( \Delta_{\text{bias}} \) based on NeuronFair, where \( \Delta_{\text{bias}} \) is the bias perturbation added to an image to flip the detection result.

First, we need a sensitive attribute classifier \( f_{\text{sat}}(x; \Theta_{sa}) \) that can distinguish the face image’s \( A_s \) (e.g., gender). We build the \( A_s \) classifier by adding a new FCN module (i.e., Fig. 4 (iii)) to the face detector’s CNN module (i.e., Fig. 4 (ii)). Then, we froze the weights of the CNN module, and train the weights of the newly added FCN module.

Next, we modify the face image’s \( A_s \) based on the adversarial attack. A classic adversarial attack FGSM [27] is adopted to flip the predicted result of the sensitive attribute by generating \( \Delta_{\text{senatt}} \) as follows:

\[
\Delta_{\text{senatt}} = \epsilon \times \text{sign}(\nabla_x J(x, y_{sa}; \Theta_{sa})) \text{ satisfying that } f_{\text{sat}}(x; \Theta_{sa}) \neq f_{\text{sat}}(x+\Delta_{\text{senatt}}; \Theta_{sa}) \quad (5)
\]

where \( \epsilon \) is a hyper-parameter to determine perturbation size, \( \text{sign}() \) is a sigmoid function return “1”, “0”, or “1”, \( x \) is an input image, \( y_{sa} \) is the ground-truth of \( x \) about sensitive attributes, \( \Theta_{sa} \) is the weights of the \( A_s \) classifier.

Finally, we leverage NeuronFair to generate \( \Delta_{\text{bias}} \), and then determine whether the instance pair \( <x+\Delta_{\text{bias}}; y_{bias}> \) satisfy Definition 1. We determine the discrimination at each layer of the detector at first. For the CNN module, the activation output of the convolutional layer is flattened. Then, in the process of image IDI generation, only the global generation is employed, which is
due to the different data forms between image and structured data. Taking the input image in Fig. 4 as an example, its attributes can be regarded as $A \in \mathbb{R}^{64 \times 64 \times 3}$. Based on a seed image IDI generated in the global phase, numerous image IDIs will evolve in the local phase. However, these image IDIs are similar, with only a few pixel differences, which have little effect on fairness improvement of the face detector. Besides, cancel the sign function $\text{sign}(\cdot)$ at line 15 of Algorithm 2 for image data.

4 EXPERIMENTAL SETTING

4.1 Datasets

We evaluate NeuronFair on 7 datasets of which five are structured datasets and two are image datasets. Each dataset is divided into three parts, i.e., 70%, 10%, 20% as training, validation, and testing, respectively.

The 5 open-source structured datasets include Adult, German credit (GerCre), bank marketing (BanMar), COMPAS, and medical expenditure panel survey (MEPS). The details of these datasets are shown in Tab. 1. All datasets can be downloaded from GitHub and preprocessed by AI Fairness 360 toolkit (AIF360) [6].

The 2 image datasets (i.e., ClbA-IN and LFW-IN) are constructed by ourselves for face detection. ClbA-IN dataset consists of 60,000 face images from CelebA [41] and 60,000 non-face images from ImageNet [17]. LFW-IN dataset consists of 10,000 face images from LFW [31] and 10,000 non-face images from ImageNet [17]. The pixel value of each image is normalized to [0,1].

4.2 Classifiers

We implement 5 FCN-based classifiers [6, 57] for structured datasets and 2 CNN-based face detectors [28, 49] for image datasets since FCN and CNN are the most widely used basic structures in real-world classification tasks.

The 5 FCN-based classifiers can be divided into two types. The one is composed of 5 hidden layers for processing low-dimensional data (i.e., Adult, GerCre, BanMar), denoted as LFCN. The another is composed of 8 hidden layers for processing high-dimensional data (i.e., COMPAS and MEPS), denoted as HFCN. The activation functions in hidden layers and the output layer are ReLU and Softmax, respectively. The hidden layer structures of LFCN and HFCN are $[64, 32, 16, 8, 4]$ and $[256, 256, 64, 64, 32, 32, 16, 8]$, respectively.

The 2 CNN-based face detectors serve for face detection, which are variants from two pre-trained models (i.e., VGG16 [49] and ResNet50 [28]) of keras.applications. We use the CNN module of VGG16 and ResNet50 as Fig. 4 (i), and design the FCN module of Fig. 4 (ii) and (iii) as $[512, 256, 128, 64, 32]$, respectively.

4.3 Baselines

We implement and compare 4 state-of-the-art (SOTA) methods with NeuronFair to evaluate their performance, including Aequitas [52], SymbGen [2], ADF [57], and EIDIG [56]. Note that Themis [24] has been shown to be significantly less effective for DNN and thus is omitted [2, 57]. We obtained the implementation of these baselines from GitHub. All baselines are configured according to the best performance setting reported in the respective papers.

4.4 Evaluation Metrics

Five aspects of NeuronFair are evaluated, including generation effectiveness, efficiency, interpretability, the utility of AUC metric, and generalization of NeuronFair.

4.4.1 Generation Effectiveness Evaluation. We evaluate the effectiveness of NeuronFair on structured data from two aspects: generation quantity and quality.

(1) Quantity. To evaluate the generation quantity, we first count the total number of IDIs, then count the global IDIs’ number and local IDIs’ number respectively, recorded as “#IDIs”. Note that the duplicate instances are filtered.

(2) Quality. We use generation success rate (GSR), generation diversity (GD), and IDIs’ contributions to fairness improvement (DM-RS [52, 56, 57]) to evaluate IDIs’ quality.

$$\text{GSR} = \frac{\# \text{IDIs}}{\# \text{non-duplicate instances}} \times 100\%$$

where non-duplicate instances represent the input space.

$$\text{GD}_{\text{NF}}(\rho_{\text{cons}}, \text{baseline}) = \frac{\text{CR}_{\text{NF}} - \text{bl}}{\text{CR}_{\text{bl}} - \text{NF}}$$

where $\text{CR}_{\text{NF}}$ and $\text{CR}_{\text{bl}}$ are the coverage rate of the NeuronFair’s IDIs to baseline’s IDIs, $\Pi_{\text{NF}}$ is the area with NeuronFair’s IDIs as the center and cosine distance $\rho_{\text{cons}}$ as the radius; similar to $\text{CR}_{\text{bl}}$.

The generated IDIs serve to improve DNN’s fairness by using these IDIs to retrain it. DM-RS is the percentage of IDIs in randomly sampled instances.

$$\text{DM-RS} = \frac{\# \text{IDIs}}{\# \text{instances randomly sampled}} \times 100\%$$

4.4.2 Efficiency Evaluation. We evaluate the efficiency of NeuronFair by generation speed [57], i.e., the time cost of generating 1,000 IDIs ($\#\text{sec}$/1,000 IDIs).

4.4.3 Interpretability Evaluation based on Biased Neurons. To interpret the utility of NeuronFair, we refer to paper [46] to design the coverage of biased neurons, which is defined as follows: for a given instance, compute the activation output of the most biased layer; normalize the activation values; select neurons with activation values greater than 0.5 as the activated neurons; compare the coverage of the activated neurons to the biased neurons.

---

*Table 1: Details of the datasets.*

| Datasets | Scenarios | Sensitive Attributes | # records | Dimensions |
|----------|-----------|----------------------|-----------|------------|
| Adult    | census income | gender, race, age | 45,842    | 13         |
| GerCre   | credit      | gender, age         | 1,000     | 20         |
| BanMar   | credit      | race                | 41,188    | 16         |
| COMPAS   | law         | medical care        | 15,675    | 137        |
| MEPS     | medical care | gender              | 5,278     | 400        |
| ClbA-IN  | face detection | gender, race     | 120,000   | $64 \times 64 \times 3$ |
| LFW-IN   | face detection | gender, race     | 20,000    | $64 \times 64 \times 3$ |

---

1https://github.com/Trusted-AI/AIF360/tree/master/aif360/data
2https://github.com/pxzhang94/ADF
3https://github.com/LingfengZhang98/EIDIG
4.4.4 Utility Evaluation of AUC Metric. In our work, based on the interpretation results, we design AUC value to measure the discrimination. We evaluate the utility of AUC metrics from three aspects: consistency, significance, and complexity between AUC and DM-RS.

(1) **Consistency.** To evaluate the consistency, we adopt Spearman’s correlation coefficient, as follows:

\[
\rho_s = 1 - \frac{6 \sum_{i=1}^{n} d_i^2}{{n}(n^2 - 1)}
\]

where \(d_i = a_i - b_i, \ i \in \{1, 2, ..., n\}, \ a_i \) and \(b_i \) are the rank of AUC and DM-RS values, respectively. High \(\rho_s\) means more consistent.

(2) **Significance.** To evaluate whether AUC can measure discrimination more significantly than DM-RS, we use the standard deviation, as follows:

\[
\sigma = \sqrt{\frac{\sum_{i=1}^{n} (c_i - \mu)^2}{n}}
\]

where \(c_i \) is AUC or DM-RS of different testing methods, \(\mu \) is the mean value of \(c_i \). Large \(\sigma\) means more significant.

4.4.5 Generalization Evaluation on Image Data. We evaluate the generalization of NeuronFair on image data from two aspects: generation quantity and quality.

(1) **Quantity.** To evaluate the generation quantity on image data, we only count the global image IDIs’ number, recorded as ‘\#IDIs’.

(2) **Quality.** We adopt GSR and IDIs’ contributions to face detector’s fairness improvement based on AUC value to evaluate IDIs’ quality, then compute its detection rate (DR) after retraining.

4.5 Implementation Details

To fairly study the performance of the baselines and NeuronFair, our experiments have the following settings: (1) the hyperparameters of each method are set according to Tab. 2, where ‘Glo.’ and ‘Loc.’ represent the global and local phases, respectively; (2) for the FCN-based classifier, we set the learning rate to 0.001, and choose Adam as the optimizer; for the CNN-based face detector, we set the learning rate to 0.01, and choose SGD as the optimizer; the training results are shown in Tab. 3, where “99.83%/92.80%/94.30%” represents the accuracy of face detector, gender classifier, and race classifier, respectively.

We conduct all the experiments on a server with one Intel i7-7700K CPU running at 4.20GHz, 64 GB DDR4 memory, 4 TB HDD and one TITAN Xp 12 GB GPU card.

---

**Table 2: Parameter setting of experiments.**

| No. | Parameters | Values (Glo. / Loc.) | Descriptions |
|-----|------------|----------------------|--------------|
| 1   | \(N_c\)    | 4 / 7                | the number of clusters for global generation |
| 2   | \(num\)   | 1,000 / 1,000        | the number of seeds for global generation |
| 3   | max_iter  | 40 / 1,000           | the maximum number of iterations for each seed |
| 4   | step_size | 1.0 / 1.0            | the perturbation size of each iteration |
| 5   | \(\mu\)   | 0.1 / 0.05           | the decay rate of momentum, \(\mu \in (0.01, 0.2)\) |
| 6   | r_step    | 10 / 50              | the step size for random disturbance, \(r_{step} \in (5, 100)\) |

**Table 3: The accuracy of classifiers and face detectors.**

| Datasets          | LFC-A | LFC-G | LFC-B | LFC-M | VGG16  | ResNet50 |
|-------------------|-------|-------|-------|-------|--------|----------|
| accuracy          | 88.36%| 100.00%| 96.71%| 92.20%| 98.13%| 99.83%/92.80%/94.30% |

**Table 4: Comparison with Aequitas, ADF, and EIDIG based on the total number of generated IDIs.**

| Datasets | Sen. Att. | Aequitas | ADF | EIDIG | NeuronFair |
|----------|-----------|----------|-----|-------|------------|
|          | #IDIs GSR | #IDIs GSR | #IDIs GSR | #IDIs GSR | #IDIs GSR |
| Adult    | gender    | age      | age | age   | gender     |
|          | 1,995 / 8.5% | 53,365 / 16.42% | 57,586 / 27.24% | 122,370 / 28.19% | 172,995 / 34.19% |
|          | 24,495 / 10.48% | 188,057 / 46.94% | 251,156 / 48.69% | 358,201 / 49.39% | 68,218 / 36.57% |
| GerCre   | gender    | age      | age | age   | gender     |
|          | 4,347 / 15.24% | 57,586 / 15.43% | 64,074 / 12.23% | 60,221 / 21.75% | 226,551 / 58.74% |
| BanMar   | gender    | age      | age | age   | gender     |
|          | 10,138 / 27.21% | 167,361 / 30.75% | 197,341 / 36.26% | 101,132 / 21.28% | 130,898 / 27.91% |
| COMPAS   | race      | age      | age | age   | gender     |
|          | 208,321 / 63.53% | 255,971 / 63.35% | 358,201 / 49.39% | 172,995 / 34.19% | 130,898 / 27.91% |

5 EXPERIMENTAL RESULTS

We evaluate NeuronFair through answering the following five research questions (RQ): (1) how **effective** is NeuronFair; (2) how **efficient** is NeuronFair; (3) how to **interpret** the utility of NeuronFair; (4) how **useful** is the AUC metric; (5) how **generic** is NeuronFair?

5.1 Research Questions 1

**How effective is NeuronFair in generating IDIs?**

When reporting the results, we focus on the following aspects: generation quantity and quality.

**Generation Quantity.** The evaluation results are shown in Tabs. 4, 5, and 6, including three scenarios: the total number of IDIs, the IDIs number in **global** phase, and the IDIs number in **local** phase.

Implementation details for quantity evaluation: (1) SymbGen works differently from other baselines, thus we follow the comparison strategy of Zhang et al. [57], i.e., evaluating the generation quantity of NeuronFair and SymbGen within the same time (i.e., 500 sec) limit, as shown in Tab. 5; (2) for a fair global phase comparison, we generate 1,000 non-duplicate instances without constrained by \(num_{it}\), then count IDIs number and record it in Tab. 6, where the seeds used are consistent for different methods; (3) for a fair local phase comparison, we mix IDIs generated globally by different methods, and randomly sample 100 as the seeds in local phase; then generate 1,000 non-duplicate instances for each seed without constrained by \(max_{iter} \), count the IDIs number on average for each seed and record it in Tab. 6.

• NeuronFair generates more IDIs than baselines, especially for densely coded structured data. For instance, in Tab. 4, on Adult dataset with different attributes, the IDIs number of NeuronFair is 217,855 on average, which is 16.5 times and 1.6 times that of Aequitas and EIDIG, respectively. In addition, in Tab. 5, NeuronFair generates much more IDIs than SymbGen on all datasets. The outstanding performance of NeuronFair is mainly because the optimization object of NeuronFair takes into account the utility of NeuronFair; (4) how useful is the AUC metric; (5) how generic is NeuronFair?
Table 5: Comparison with SymbGen based on the number of IDIs generated in 500 seconds.

| Datasets          | Sen. Att. | SymbGen | Aequitas | NeuronFair |
|-------------------|-----------|---------|----------|------------|
|                   | #IDIs     | GSR     | #IDIs    | GSR        |
| Adult gender      | 195       | 13.89%  | 4,048    | 25.24%     |
| Adult race        | 452       | 11.01%  | 4,532    | 39.54%     |
| Adult age         | 531       | 12.17%  | 3,610    | 27.55%     |
| Adult age         | 1,034     | 37.19%  | 5,760    | 50.74%     |
| GerCre gender     | 821       | 18.92%  | 3,252    | 26.35%     |
| GerCre age        | 672       | 30.79%  | 2,264    | 35.00%     |
| BanMar race       | 2,264     | 35.00%  | 5,760    | 50.74%     |
| BanMar age        | 3,252     | 26.35%  | 4,048    | 25.24%     |
| COMPAS race       | 84        | 2.32%   | 2,264    | 35.00%     |
| COMPAS age        | 42        | 1.33%   | 426      | 6.09%      |
| MEPS gender       | 404       | 14.22%  | 3,796    | 51.40%     |
| MEPS age          | 92        | 2.32%   | 864      | 14.45%     |

For instance, on all datasets, the IDIs number of NeuronFair is 866 on average, which is 9.45 times and 1.42 times that of Aequitas and EIDIG, respectively. This is mainly because the optimization object of NeuronFair takes into account the dynamics through the dynamic combination of biased neurons. Thus, NeuronFair searches a larger space to generate more global IDIs.

- In the local phase, NeuronFair is much more efficient than baselines in general. For instance, in Tab. 6, on average, NeuronFair returns 78.97%, 45.35%, 10.81%, and 2.90% more IDIs than Aequitas, SymbGen, ADF, and EIDIG, respectively. Recall that Aequitas, ADF, EIDIG, and NeuronFair all guide local phase through a probability distribution, which is the likelihood of IDIs by modifying several certain attributes (i.e., the loop from lines 11 to 16 of Algorithm 3). The probability determination of NeuronFair takes into account the momentum and SoftMax activation (i.e., at line 10 of Algorithm 3) while the baselines do not. Hence, NeuronFair generates more local IDIs.

**Generation Quality.** The evaluation results are shown in Tabs. 4, 5, 7, and Fig. 5, including the generation success rate (GSR), generation diversity (GD), and fairness improvement (DM-RS).

Implementation details for quality evaluation: (1) for a fair diversity comparison, we seed each method with the same set of 10 seeds and record the average DM-RS value in Tab. 7.

For instance, in Tab. 4, on the GerCre dataset with gender and age attributes, the GSR values of NeuronFair are 36.57% and 63.35%, whereas that of Aequitas is 17.23% and 59.38%. We speculate the reason is that the discrimination about the gender attribute in the output layer is not obvious, but NeuronFair can find potential fairness violations through the internal discrimination information of biased neurons. Therefore, we can realize stable testing for different sensitive attributes.

In addition, the valid input space of NeuronFair is larger than baselines in general, i.e., a larger input space supports more diverse IDI generation. For instance, in Tab. 5, the average input space of NeuronFair is 3.30 times that of SymbGen. It is mainly because the momentum acceleration strategy employs historical gradient as auxiliary guidance, which reduces the number of invalid searches. Hence, NeuronFair generates more IDIs in a large input space.

- In all cases, NeuronFair can generate more diverse IDIs, which is beneficial to discover more potential discrimination and then improve fairness through retraining. For instance, in Fig. 5, compare to Aequitas and ADF, the GDS values are all greater than 1 under different radius values $\rho_{cons}$, and as the radius increases, the value of GDS gradually converges to 1. It demonstrates that the IDIs generated by NeuronFair can always cover that of Aequitas. The outstanding performance of NeuronFair is mainly because it not only considers the whole DNN’s discrimination through biased neurons, but also takes into account the dynamics of the optimization object through the combination of biased neurons. Thus, NeuronFair searches a larger valid input space than Aequitas.

Meanwhile, the GSR value of NeuronFair on different sensitive attributes is more robust than ADF and EIDIG. For instance, in Tab. 4, on the GerCre dataset with gender and age attributes, the GSR values of NeuronFair are 36.57% and 63.35%, whereas that of EIDIG are 17.23% and 59.38%. We speculate the reason is that the discrimination about the gender attribute in the output layer is not obvious, but NeuronFair can find potential fairness violations through the internal discrimination information of biased neurons. Therefore, we can realize stable testing for different sensitive attributes.

In addition, the valid input space of NeuronFair is larger than baselines in general, i.e., a larger input space supports more diverse IDI generation. For instance, in Tab. 5, the average input space of NeuronFair is 3.30 times that of SymbGen. It is mainly because the momentum acceleration strategy employs historical gradient as auxiliary guidance, which reduces the number of invalid searches. Hence, NeuronFair generates more IDIs in a large input space.

- In all cases, NeuronFair can generate more diverse IDIs, which is beneficial to discover more potential discrimination and then improve fairness through retraining. For instance, in Fig. 5, compare to Aequitas and ADF, the GDS values are all greater than 1 under different radius values $\rho_{cons}$, and as the radius increases, the value of GDS gradually converges to 1. It demonstrates that the IDIs generated by NeuronFair can always cover that of baselines. We speculate the reason is that the dynamic loss function expands the valid input space by combining different biased neurons as the optimization object.

![Figure 5: Generation diversity of NeuronFair compared to Aequitas (left) and ADF (right).](Image)
### Table 8: Time (sec) taken to generate 1,000 IDIs.

| Datasets | Sen. | Att. | Acquitas | SymbGen | ADF | EIDIG | NeuronFair |
|----------|-----|-----|---------|---------|----|------|------------|
| Adult    | age | 345.68 | 1,568.20 | 298.46 | 156.38 | 534.56% | 121.56 |
| GerCre   | age | 346.00 | 2,014.68 | 488.19 | 344.10 | 294.66 |
| BanMar   | age | 597.00 | 3,035.21 | 472.59 | 246.64 | 116.52 |
| COMPAS   | race | 524.13 | 2,315.94 | 253.69 | 199.93 | 187.50 |
| MEPS     | gender | 498.16 | 2,537.58 | 213.76 | 118.85 | 103.91 |

Besides, a close investigation shows that there is a similar trend in the generation diversity for the same sensitive attribute. For instance, when $p_{cons} < 0.1$ in Fig. 5 (a) or $p_{cons} < 0.02$ in Fig. 5 (b), the line ‘L2’ with race is always the highest, the line ‘L4’ with age is always the lowest, while lines ‘L1’ and ‘L3’ with gender are in the middle. Since both datasets Adult and GerCre are related to money (i.e., salary and loans), we speculate that there is similar discrimination for gender in classifiers LFC-A and LFC-G for similar tasks, thus $GD_{NF}$ shows similar trends in gender attribute.

- In all cases, NeuronFair can obtain larger DM-RS values, i.e., the IDIs generated by NeuronFair contribute more to the DNNs’ fairness improvement. For instance, in Tab. 7, measured by DM-RS, NeuronFair realizes fairness improvement of 87.24% on average, versus baselines, i.e., 81.18% for Aequitas, 80.78% for SymbGen, 83.30% for ADF, and 84.49% for EIDIG. It is because the IDIs of NeuronFair are more diverse than those of baselines, so it can discover more potential fairness violations and implement higher fairness improvement through retraining.

**Answer to RQ1:** NeuronFair outperforms the SOTA methods (i.e., Aequitas, SymbGen, ADF, and EIDIG) in two aspects: (1) quantity - it generates $\sim 5.84$ IDIs on average compared to baselines; (2) quality - it searches $\sim 3.03$ input space with more than $\sim 1.65$ GSR on average compared to baselines, it generates IDIs that are $\sim 6.24$ and $\sim 1.38$ more diverse than Aequitas and ADF on average with $p_{cons} < 0.02$, it is beneficial to DNNs’ fairness improvement of 87.24% on average.

### 5.2 Research Questions 2

**How efficient is NeuronFair in generating IDIs?**

When answering this question, we refer to the generation speed. The evaluation results are shown in Tab. 8, where the time cost of SymbGen includes generating the explainer and constraint solving. Here we have the following observation.

- NeuronFair generates IDIs more efficiently, which meets the rapidity requirements of software engineering testing. For instance, in Tab. 8, on average, NeuronFair takes only 26.07%, 5.47%, 49.47%, and 79.32% of the time required by Aequitas, SymbGen, ADF, and EIDIG, respectively. The outstanding performance of NeuronFair is mainly because it uses a momentum acceleration strategy and shortens the derivation path to reduce computational complexity. Hence, it takes less time than baselines.

**Answer to RQ2:** NeuronFair is more efficient in generation speed - it produces IDIs with an average speedup of 534.56%.

### 5.3 Research Questions 3

**How to interpret NeuronFair’s utility by biased neurons?**

When interpreting the utility, we refer to the biased neuron coverage. The evaluation results are shown in Fig. 6.

Implementation details for interpretation: (1) we conduct experiments on the Adult dataset with gender attribute for the LFC-A classifier; (2) we compare the interpretation results of NeuronFair with ADF and EIDIG; (3) for a fair interpretation, we randomly select 10% IDIs and 10% non-IDIs (i.e., the generated failure instances) for each method, and then compute the coverage of biased neurons, as shown in Fig. 6.

- Biased neurons can be adopted to interpret the utility of IDIs and NeuronFair. First, IDIs trigger discrimination by activating biased neurons. For instance, the neurons activated by IDIs can cover most of the biased neurons in Fig. 6 (a), while the coverage of the biased neurons by non-IDIs of different methods is 0 in Fig. 6 (b). We can further interpret the utility of testing methods is related to the coverage of biased neurons, i.e., NeuronFair is more effective than ADF and EIDIG because they miss some discrimination contained in biased neurons while NeuronFair does not. For instance, in Fig. 6 (a), the NeuronFair’s IDIs activate all 24 biased neurons in the 2-nd layer of LFC-A classifier, while the neurons activated by other IDIs cannot cover all (15 for ADF and 18 for EIDIG).

**Answer to RQ3:** The main reason for NeuronFair’s utility is that its IDIs can activate more biased neurons. NeuronFair’s IDIs activate 100% biased neurons, while 62.5% for ADF and 75% for EIDIG.

### 5.4 Research Questions 4

**How useful is the AUC metric for measuring DNNs’ fairness?**

When answering this question, we refer to the following aspects: the consistency, significance, and complexity between AUC and DM-RS. The evaluation results on Adult and GerCre datasets with multiple sensitive attributes are shown in Tab. 9. From the results, we have the following observations.

- In all cases, AUC can correctly distinguish DNNs’ fairness violations, i.e., AUC can serve the discrimination measurement of DNN. For instance, in Tab. 9, all of the $p_s$ values are 1.00, indicating that the discrimination ranking results of different DNNs based on AUC are completely consistent with those based on DM-RS. Since DNN’s decision results are determined by the neurons’ activation, we speculate that the biased decisions are also caused by the neurons’ activation, i.e., neurons contain discrimination information. Therefore, we can leverage the discrimination information in neurons to determine DNNs’ fairness.
Table 9: The consistency and significance between DM-RS and AUC (ρ, σ). Note that here we only use normal instance pairs to compute each layer’s AUC, and select the maximum AUC as the classifier’s discrimination.

| Dataset | Sen. Att. | Metrics | Before | After | AUC | ADIF | EIDIG | NeuronFair |
|---------|-----------|---------|--------|-------|-----|-------|-------|------------|
|       |           |         |        |       |     |       |       |            |
| Adult  | gender    | DM-RS   | 3.28%  | 2.81% | 0.753 | 0.1922 | 0.1482 | 0.1331 | 0.1098 | 0.0897 | 0.2563 |
|        | race      | DM-RS   | 8.91%  | 5.15% | 0.8045 | 0.1466 | 0.1595 | 0.1652 | 0.1599 | 0.1070 | 0.2677 |
| GerCre | gender    | DM-RS   | 14.56% | 10.46%| 0.8391 | 0.1566 | 0.1520 | 0.1482 | 0.1347 | 0.1082 | 0.2941 |
|        | age       | DM-RS   | 30.30% | 25.16%| 0.8608 | 0.2046 | 0.1691 | 0.1568 | 0.1432 | 0.1204 | 0.2879 |

Table 10: ‘IDIs’ and ‘GSR’ measurements in the global phase on image datasets.

| Datasets | Sen. Att. | ADF | EIDIG | NeuronFair |
|----------|-----------|-----|-------|------------|
|          |           |     |       |            |
| CIbA-IN  | gender    | 1.087 | 11.58% | 2.895 | 12.50% | 10.578 | 69.90% |
|          | race      | 11.908 | 33.54% | 25.180 | 59.87% | 51.529 | 90.15% |
| LPW-IN   | gender    | 1.204 | 33.20% | 1.105 | 40.10% | 3.950 | 61.40% |
|          | race      | 2.269 | 31.70% | 5.304 | 62.40% | 5.457 | 64.17% |

Table 11: Fairness improvement of face detectors.

| Datasets | Sen. Att. | ADF | EIDIG | NeuronFair |
|----------|-----------|-----|-------|------------|
|          |           |     |       |            |
| CIbA-IN  | gender    | 0.3587 | 99.83% | 0.3328 | 97.20% | 0.3091 | 95.40% |
|          | race      | 0.4438 | 99.83% | 0.4045 | 96.58% | 0.3720 | 95.50% |
| LPW-IN   | gender    | 0.3900 | 99.56% | 0.3324 | 96.58% | 0.3068 | 92.30% |
|          | race      | 0.4251 | 99.36% | 0.3984 | 98.10% | 0.3933 | 96.40% |

- In all cases, AUC can distinguish the different DNN’s discrimination more significantly than DM-RS, which is beneficial for a more accurate evaluation of IDIs of different testing methods. For instance, in Tab. 9, all σ values of AUC are higher than those of DM-RS, and the average σ value of AUC is 8.46 times that of DM-RS. The outstanding performance of AUC is mainly because we use the neurons’ ActDiff to measure the discrimination, which extracts more bias-related information from the whole DNN; while MD-RS only uses the bias-related information from the output layer.

- In all cases, AUC can distinguish the different DNN’s discrimination more significantly than DM-RS, which is beneficial for a more accurate evaluation of IDIs of different testing methods. For instance, in Tab. 9, all σ values of AUC are higher than those of DM-RS, and the average σ value of AUC is 8.46 times that of DM-RS. The outstanding performance of AUC is mainly because we use the neurons’ ActDiff to measure the discrimination, which extracts more bias-related information from the whole DNN; while MD-RS only uses the bias-related information from the output layer.

- The computational complexity of AUC is much lower than that of MD-RS, which is beneficial to quickly distinguish DNNs’ discrimination or the IDIs’ effect. The time frequency of AUC is T(N) = (7 + count)×N_N+1 based on Algorithm 1. Thus the time complexity of AUC is O(N_N), while that of DM-RS is O(n log n), where n is the instance number, N_N is the layer number, n >> N_N. It is mainly because AUC only conducts matrix operations, while DM-RS requires iterative operations until convergence.

**Answer to RQ4**: The AUC is useful for discrimination measurement. Compared to the results in Tab. 9, AUC is (1) 100% consistent with DM-RS, (2) ×8.46 more significant than DM-RS, (3) low computational complexity with O(N_N).

5.5 Research Questions

How generic is NeuronFair for the task of image IDI generation?

When reporting the results, we focus on two aspects: generation quantity and quality.

Implementation details for NeuronFair generalized on image data: (1) we only perform comparisons with ADF and EIDIG at global phase, because the effect of ADF and EIDIG on DNNs is much better than that of Aequitas and SymbGen; (2) we remove the KMeans() operation, set step-size=0.15 for image, and all face images are used as input; (3) we retrain the face detector with all image IDIs of each method and measure its fairness improvement by AUC, (4) we measure the bias perturbation Δbias and sensitive attribute perturbation Δsenatt by L2-norm.

**Generation Quantity**: The evaluation results are shown in Tab. 10 measured by the IDIs number in global phase. From the results, we have the following observation.

- In all cases, NeuronFair can obtain more IDIs than ADF and EIDIG, especially for the discrimination against race attribute. For instance, in Tab. 10, on average, NeuronFair generates 4.34 times and 2.07 times IDIs of ADF and EIDIG, respectively. The outstanding performance of NeuronFair is because it adopts dynamic loss to expand the valid input space while ADF and EIDIG do not consider the dynamics of search. Meanwhile, the number of IDIs generated by NeuronFair for race is 3.91 times that for gender. We speculate the reason is that the pixel information related to race is mainly skin color (i.e., light & dark, or black & white), while the pixel information related to gender is more diverse (such as hair, makeup, face shape, etc.). Therefore, the image IDIs generation for race is easier through manipulating skin color.

**Generation Quality**: The evaluation results are shown in Tabs. 10 and 11, including three scenarios: the generation success rate (GSR), the fairness improvement (AUC), and the detection rate (DR).

- Among image data, image IDIs of NeuronFair are of higher quality than those of ADF and EIDIG, which can be applied to retrain face detectors and contribute to their fairness improvement in face detection scenarios. For instance, in Tab. 10, on average, the GSR value of NeuronFair is 2.60 times and 1.63 times that of ADF and EIDIG, respectively. It is because NeuronFair reduces the probability of gradient vanishing, which in turn improves the probability of non-duplicate IDIs generation guided by the gradient. Hence, all GSR values of NeuronFair are higher than those of baselines. Meanwhile, the valid input space of NeuronFair is 1.67 times and 1.27 times that of ADF and EIDIG, respectively. Since the probability of falling into a local optimum is reduced by dynamically combining biased neurons, we can perform valid searches in limited instance space.

- NeuronFair contributes more to the fairness improvement of the face detector, i.e., its generalization on image data is better than that of ADF and EIDIG. For instance, in Tab. 11, on average, the discrimination of detectors retrained with IDIs of NeuronFair dropped by 53.77%, while the AUC values of ADF and EIDIG only dropped by 8.06% and 10.90%, respectively. We speculate that the valid input space of NeuronFair is larger, so its IDIs can find potential discrimination that other methods’ IDIs cannot. Then improve the detector’s fairness through retraining.

- NeuronFair hardly affects the detector’s DR values while improving its fairness. For instance, in Tab. 11, on average, the DR value of detectors retrained with NeuronFair’s IDIs only dropped by
0.87%, while that of ADF and EIDIG dropped by 2.92% and 4.80%, respectively. We compare the $L_2$ norm of $\Delta_{bias}$ and $\Delta_{senate}$ generated by different methods, and find that $\Delta_{bias}$ of NeuronFair is much lower than that of ADF and EIDIG. Therefore, NeuronFair can not only improve the detector’s fairness but also maintain its detection performance.

**Answer to RQ5:** The generalization performance of NeuronFair on the image dataset is better than the SOTA methods (i.e., ADF and EIDIG) in two aspects: (1) *quantity* - it generates $\times 4.34$ and $\times 2.07$ image IDIs on average compared to ADF and EIDIG, respectively; (2) *quality* - it searches $\times 1.47$ input space with more than $\times 2.11$ GSR on average, it is beneficial to detectors’ fairness improvement of 53.77% on average but hardly affects their detection performance. Thus, NeuronFair shows better generalization performance than ADF and EIDIG.

**6 THREATS TO VALIDITY**

**Correlation between attributes.** The attributes of unstructured data are not as clear as structured data, so we provide a generalization framework that can modify sensitive attributes. However, there is a correlation between attributes, i.e., after the perturbation for one sensitive attribute is added, another attribute may also be changed. Since the transferability of perturbation is not robust, the slight attribute change will not affect our IDI generation.

**Sensitive attributes.** We consider only one sensitive attribute at a time for our experiments. However, considering multiple protected attributes will not hamper the effectiveness or generalization offered by our novel testing technique, but will certainly lead to an increase in execution time. This increase is attributed towards the fact that the algorithm in such a case, needs to consider all the possible combinations of their unique values.

**Access to DNNs.** NeuronFair is white-box testing that generates IDIs based on the biased neurons inside DNNs, which means it requires accessing to DNNs. It is widely accepted that DNN testing could have full knowledge of the target model in software engineering.

**7 RELATED WORKS**

**Fairness Testing.** Based on the software engineering point of view, several works on testing the fairness of traditional ML models are proposed [1, 2, 24, 51, 52, 55]. To uncover their fairness violations, Galhotra et al. [24] firstly proposed Themis, a fairness testing method for software, which measures the discrimination in software through counting the frequency of IDIs in the input space. However, its efficiency for IDIs generation is unsatisfactory. To improve the generation speed of Themis, Udeshi et al. [52] proposed a faster generation algorithm, Aequitas, which uncovers fairness violations by probabilistic search over the input space. Aequitas adopts a two-phase operation in which the IDIs generated globally are used as seeds for the local generation. However, Aequitas uses a global sampling distribution for all the inputs, which leads to the limitation that it can only search in narrow input space and easily falls into the local optimum. Thus, Aequitas’s IDIs lack diversity. To further improve the instance diversity, Agarwal et al. [2] designed a new testing method, SymbGen, which combines the symbolic execution along with the local interpretation for the generation of effective instances. SymbGen constructs the local explainer of the complex model at first and then searches for IDIs based on the fitted decision boundary. Therefore, its instance effectiveness almost depends on the performance of the explainer.

The above-mentioned methods mainly deal with traditional ML models, which cannot directly be applied to deal with DNNs. Recently, several methods have been proposed specifically for DNNs. For instance, Zhang et al. [57] first proposed a fairness testing method specifically for DNNs, ADF, which guides the search direction through gradients. The authors proved that its effectiveness and efficiency of IDIs generation for DNNs are greatly improved based on the guidance of gradients. Based on the ADF [57], Zhang et al. [56] designed a framework EIDIG for discovering individual fairness violations, which adopts prior information to accelerate the convergence of iterative optimization. However, there is still a problem of gradient vanishing, which may lead to local optimization.

**Neuron-based DNN Interpretation.** Kim et al. [33] first introduced concept activation vectors, which provide an interpretation of a DNN’s internal state (i.e., the activation output in the hidden layer). They viewed the high-dimensional internal state of a DNN as an aid, and interpreted which concept is important to the classification result. Inspired by the concept activation vectors, Du et al. [21] suggested that interpretability can serve as a useful ingredient to diagnose the reasons that lead to algorithmic discrimination. The above methods study the activation output of one hidden layer, while Liu et al. [40] studied the activation state of a single neuron. They observed that the neuron activation is related to the DNNs’ robustness, and used the abnormal activation of a single neuron to detect backdoor attacks. These methods leverage the internal state to interpret DNNs’ classification performance and robustness, which inspires us to use it to interpret DNNs’ biased decision.

**8 CONCLUSIONS**

We propose an interpretable white-box fairness testing method, NeuronFair, to efficiently generate IDIs for DNNs based on biased neurons. Our method provides discrimination interpretation and IDI generation for different data forms. In the discrimination interpretation, AS curve and AUC measurement are designed to qualitatively and quantitatively interpret the severity of discrimination in each layer of DNNs, respectively. In the IDI generation, a global phase and a local phase collaborate to systematically search the input space for IDIs with the guidance of momentum acceleration and dynamic loss. Further, NeuronFair can process not only structured data but also unstructured data, e.g., image, text, etc. We compare NeuronFair with four SOTA methods in 5 structured datasets and 2 face image datasets against 7 DNNs, the results show that NeuronFair has significantly better performance in terms of interpretability, generation effectiveness, and data generalization.

**ACKNOWLEDGMENT**

This research was supported by the National Natural Science Foundation of China (Nos. 62072406, 62102359), the Key R&D Projects in Zhejiang Province (Nos. 2021C01117 and 2022C01018), the “Ten Thousand Talents Program” in Zhejiang Province (2020R52011).
