Information-Theoretic Testing and Debugging of Fairness Defects in Deep Neural Networks

Verya Monjezi
vmonjezi@miners.utep.edu
University of Texas at El Paso

Ashutosh Trivedi
ashutosh.trivedi@colorado.edu
University of Colorado Boulder

Gang Tan
gtan@psu.edu
Pennsylvania State University

Saeid Tizpaz-Niari
saeid@utep.edu
University of Texas at El Paso

***Abstract*—The deep feedforward neural networks (DNNs) are increasingly deployed in socioeconomic critical decision support software systems. DNNs are exceptionally good at finding minimal, sufficient statistical patterns within their training data. Consequently, DNNs may learn to encode decisions—amplifying existing biases or introducing new ones—that may disadvantage protected individuals/groups and may stand to violate legal protections. While the existing search based software testing approaches have been effective in discovering fairness defects, they do not supplement these defects with debugging aids—such as severity and causal explanations—crucial to help developers triage and decide on the next course of action. Can we measure the severity of fairness defects in DNNs? Are these defects symptomatic of improper training or they merely reflect biases present in the training data? To answer such questions, we present DICE: an information-theoretic testing and debugging framework to discover and localize fairness defects in DNNs.**

The key goal of DICE is to assist software developers in triaging fairness defects by ordering them by their severity. Towards this goal, we quantify fairness in terms of protected information (in bits) used in decision making. A quantitative view of fairness defects not only helps in ordering these defects, our empirical evaluation shows that it improves the search efficiency due to resulting smoothness of the search space. Guided by the quantitative fairness, we present a causal debugging framework to localize inadequately trained layers and neurons responsible for fairness defects. Our experiments over ten DNNs, developed for socially critical tasks, show that DICE efficiently characterizes the amounts of discrimination, effectively generates discriminatory instances (vis-a-vis the state-of-the-art techniques), and localizes layers/neurons with significant biases.

I. INTRODUCTION

AI-assisted software solutions—increasingly implemented as deep neural networks [1] (DNNs)—have made substantial inroads into critical software infrastructure where they routinely assist in socio-economic and legal-critical decision making [2]. Instances of such AI-assisted software include software deciding on recidivism, software predicting benefit eligibility, and software deciding whether to audit a given taxpayer. The DNN-based software development, driven by the *principle of information bottleneck* [3], involves a delicate balancing act between over-fitting and detecting useful, parsimonious patterns. It is, therefore, not a surprise that such solutions often encode and amplify pre-existing biases in the training data. What's worse, improper training may even introduce biases not present in the training data or irrelevant to the decision making. The resulting fairness defects may not only disadvantage protected groups [4], [5], [6], [7], [8], but may stand to violate statutory requirements [9].

> *This paper presents* DICE*, an information-theoretic testing and debugging framework for fairness defects in deep neural networks.*

Quantifying Fairness. Concentrated efforts from the software engineering and the machine learning communities have produced a number of successful *fairness testing* frameworks [10], [11], [12], [13]. These frameworks characterize various notions of fairness—such as group fairness [14] (decision outcome for various protected groups must be similar) and individual fairness [15] (individuals differing only on protected attributes must receive similar outcome)—and employ search-based testing to discover fairness defects. While a binary classification of fairness is helpful in discovering defects, developers may require further insights into the nature of these defects to decide on the potential "bug fix". Are some defects more severe than others? Whether these defects stem from biases present in the training data, or they are artifacts of an inadequate training? Is it possible to find an alternative explanation of the training data that does not use protected information?

Individual discrimination is a well-studied [16], [11], [17], [18] causal notion of fairness that defines a function being discriminant towards an individual (input) if there exists another individual (potentially counterfactual), differing only in the protected features, receives a more favorable outcome. We present a quantitative generalization of this notion as the *quantitative individual discrimination* (QID). We define QID as the amount of protected information—characterized by entropy metrics such as Shannon entropy and min entropy—used in deriving an outcome. Observe that a zero value for the QID measure implies the absence of the individual discrimination. The QID measure allows us to order various discriminating inputs in terms of their severity, as in an application that is not supposed to base its decisions on protected information, inputs with higher dependence indicate a more severe violation. Our first *research question* (**RQ1**) concerns the usefulness of QID measure in finding inputs with different severity.

Search-Based Testing. Search-based software testing provide scalable optimization algorithms to automate discovery of software bugs. In the context of fairness defects, the search of such bugs involves finding twin inputs exhibiting discriminatory instances. The state-of-the-art algorithms for fairness testing [19], [17], [18] explore the input space governed by a binarized feedback, resulting in a discontinuous search domain. On the other hand, QID-based search algorithms can benefit a smooth (quantitative) feedback during the optimization, resulting in a more guided search. Our next research question (**RQ2**) is to investigate whether this theoretical promise materializes in practice in terms of discovering richer discriminating instances than using classic notions of discrimination.

Causal Explanations. While the discriminating instances (ordered by their severity) provide a clear evidence of fairness defects in the DNN, it is unclear whether these defects are inherent in the training data, or whether they are artifacts of the training process. Inspired by the notion of "the average causal effects" [20] and AUDEE framework [21] for bug localization in deep learning models, we develop a layer and neuron localization framework for fairness defects. If the cause of the defects is found to be at the input layer, it is indicative of discrimination existing in the training data. On the other hand, if we localize the cause of the defect to some internal layer, we wish to further prod the DNN to extract quantitative information about neurons and their counterfactual parameters that can mitigate the defect while maintaining the accuracy. This debugging activity informed our next research question (**RQ3**): is it possible to identify a subset of neurons and their causal effects on QID to guide a mitigation without affecting accuracy?

Experiments. DICE implements a search algorithm (Algorithm 1) to discover inputs that maximize QID and a causal debugging algorithm (Algorithm 2) to localize layers and neurons that causally affect the amounts of QID. Using 10 socio-critical DNNs from the literature of algorithmic fairness, we show that DICE finds inputs that can use significant amounts of protected information in the decision making; outperforms three state-of-the-art techniques [19], [17], [18] in generating discriminatory instances; and localizes neurons that guides a simple mitigation strategy to reduce QID down to 15% of reported initial QID with at most 5% loss of accuracy. The key contributions of this paper are:

1) **Quantitative Individual Discrimination.** We introduce an information-theoretic characterization of discrimination, dubbed quantitative individual discrimination (QID), based on Shannon entropy and Min entropy.
2) **Search-based Testing.** We present a search-based algorithm to discover circumstances under which the DNNs exhibit severe discrimination.
3) **Causal Debugging.** We develop a causal fairness debugging based on the language of interventions to localize the root cause of the fairness defects.
4) **Experimental Evaluation.** Extensive experiments over different datasets and DNN models that show feasibility, usefulness, and scalability (viz-a-viz state-of-the-art). Our framework can handle multiple protected attributes and can easily be adapted for regression tasks.

II. PRELIMINARIES

Fairness Terminology. We consider decision support systems as *binary classifiers* where a prediction label is *favorable* if it gives a desirable outcome to an input (individual). These favorable predictions may include higher income estimations for loan, low risk of re-offending in parole assessments, and high risk of failing a class. Each dataset consists of a number of *attributes* (such as income, experiences, prior arrests, sex, and race) and a set of *instances* that describe the value of attributes for each individual. According to ethical and legal requirements, data-driven software should not *discriminate* on the basis of an individual's *protected attributes* such as sex, race, age, disability, colour, creed, national origin, religion, genetic information, marital status, and sexual orientation.

There are several well-established fairness definitions. *Group fairness* requires that the statistics of ML outcomes for different *protected groups* to be similar [14] using metrics such as *equal opportunity difference* (EOD), which is the difference between the true positive rates (TPR) of two protected groups. Fairness through unawareness (FTU) [15] requires removing protected attributes during training. However, FTU may provide inadequate support since protected attributes can influence the prediction via a non-protected collider attribute (e.g., race and ZIP code). Fairness through awareness (FTA) [15] is an *individual fairness* notion that requires that two *individuals* that deemed similar (based on their non-protected attributes) are treated similarly. *Our approach is geared toward individual fairness.*

Individual Discrimination. Causal discrimination, first studied in THEMIS [10], measures the difference between two subgroups via *counterfactual* queries. It samples individuals with the protected attributes set to A and compares the outcome to a counterfactual scenario where the protected attributes is set to B. Individual discrimination (ID) is a prevalent notion that adapts counterfactual queries to find an individual such that their counterfactual with a different protected attributes receives more favorable outcome. This fairness notion is used by the state-of-the-art fairness testing to generate fairness defects [22], [19], [17], [18] and closely related to situation testing notion [23]. While standard group fairness metrics (e.g., AOD/EOD) are already quantitative, the quantitative measures do not exist for individual fairness. We propose to adapt information theoretic tools to provide quantitative measures for individual fairness.

Information-Theoretic Concepts. The notion of Rényi entropy [24], $H_\alpha(X)$ quantifies the uncertainty (randomness) of a system responding to inputs X. In particular, Shannon entropy (α=1) and min-entropy (α=∞) are two important subclasses of Rényi entropy. Shannon entropy (H_1) measures the expected amounts of uncertainty over finitely many events whereas min entropy (H_∞) measures the uncertainty over single maximum likelihood event.

Consider a deterministic system (like pre-trained DNN) with a finite set of responses and assume that the input X is distributed uniformly. Thus, the system induces an equivalence relation over the input set X such that two inputs are equivalent if their system outputs are approximately close, i.e. $x \sim x'$ iff $DNN(x) \approx_\epsilon DNN(x')$. Let X_o denote the equivalence class of X with output o. Then, the remaining uncertainty after observing the output of DNN over X can be written as:

$$H_1(X|O) = \sum_{O=o} \frac{|X_o|}{|X|}.\log_2(|X_o|) \qquad \text{(Shannon entropy)}$$

where $|X|$ is the cardinality of X and $|X_o|$ is the size of equivalence class of output o. Similarly, the min-entropy is given as

$$H_\infty(X|O) = \log_2(\frac{|X|}{|O|}) \qquad \text{(min-entropy)}$$

where $|O|$ is the number of equivalence classes over X [25], [26], [27]. Given that the initial entropy is equal to $\log_2(|X|)$ for both entropies, the amount of information from X used by the system to make decisions are

$$\begin{aligned} I_1(X;O) &= \log_2(|X|) - H_1(X|O), \text{ and} \\ I_\infty(X;O) &= \log_2(|O|), \end{aligned}$$

under Shannon- and min- entropies with $I_1 \leq I_\infty$.

Quantitative Notion and Fairness. Our approach differs from these state-of-the-art techniques [28], [22], [19], [17], [29], [18] in that it extends the individual discrimination notion with quantitative information flow that enables us to measure the amount of discrimination in the ML-based software systems. Given non-protected attributes and ML outcomes, the *Shannon entropy* measures the expected amount of individual discrimination over all possible responses varying protected classes, whereas, *min entropy* measures the amount over a single response from the maximum likelihood class.

Example 1. *Consider a dataset with* 16 *different protected values—sex(2), race(2), and age(4)—distributed uniformly, and suppose that we have* 4 *individuals in the system. We perturb the protected attributes of these individuals to generate* 16 *counterfactuals and run them through the DNN to get their prediction scores. Suppose that the outputs have all* 16 *outputs in the same class for the first individual (absolutely fair) and have* 16 *classes of size one for the second individual (absolutely discriminatory). For the third individual, let the outputs be in* 4 *classes with* $\{4,4,4,4\}$ *elements in each class (e.g., there is one output class per each age group). For the fourth individual, let us consider outputs to be in* 5 *classes with* $\{8,4,2,1,1\}$ *elements in each class (e.g.,* `if` *race=1* `then` *the output is class* 1*; else,* `if` *sex=1* `then` *the output is* 2*; else* `if` *age=*$\{1,2\}$ `then` *the output is* 3*; else there is one output class for each age=*$\{3,4\}$*).*

We work with the following notions of discrimination.

- *Individual Discrimination Notion*. The individual discrimination used by the state-of-the-art techniques can only distinguish between the first individual and the rest, but they cannot distinguish among individuals two to four. In fact, these techniques generate tens of thousands of individual discriminatory instances in a short amount of time [19], [17], [18]. However, they fail to prioritize test cases for mitigation and cannot characterize the amounts of discrimination (i.e., their severity).
- *Shannon Entropy*. Using the Shannon entropy, we have the initial fairness to be 4.0 bits, a maximum possible discrimination. The remaining fairness of DNN are 4.0, 0.0, 2.0 and 2.125, for the first to fourth individuals, respectively. The discrimination is the difference between the initial and remaining fairness, which are 0.0, 4.0, 2.0 and 1.875 for the first to fourth individuals, respectively. It is important to note that beyond the two extreme cases, Shannon entropy deems perturbations to the third individual (rather than the fourth) create a higher amount of discrimination.
- *Min Entropy*. The initial fairness via min entropy is also 4 bits. The conditional min entropy is $\log \frac{16}{1} = 4.0$, $\log \frac{16}{16} = 0.0$, $\log \frac{16}{4} = 2.0$, and $\log \frac{16}{5} = 1.7$, for the four individuals, respectively. The amounts of discrimination thus are 0.0, 4.0, $\log 4 = 2.0$ and $\log 5 = 2.3$, respectively. Beyond the two extreme cases where both entropies agree, the min entropy deems perturbations to the fourth individual create a higher amount of discrimination. This is intuitive since the discrimination on the fourth case is more subtle, complex, and significant. Therefore, the ML software developers might prioritize those cases characterized by the min entropy.

III. Overview

DICE in Nutshell. Figure 1 shows an overview of our framework DICE. It consists of two components: (1) an automatic test-generation mechanism based on search algorithms and (2) a debugging approach that localizes the neurons with significant impacts on fairness using a causal algorithm. First, DICE searches through the space of input dataset to find circumstances on the non-protected attributes under which the DNN-under-test shows a significant dependency on the protected attributes in making decisions. In doing so, it works in global and local phases. In the global phase, the search explores the input space to increase the amount of discrimination in each step of search. On the other hand, in the local phase it exploits the promising seeds from the global phase to generate as many discriminatory instances as possible.

The key elements of search is a threshold-based clustering algorithm used for computing both gradients and objective functions that provide a smooth feedback. The search characterizes the quantitative individual discrimination (QID) and

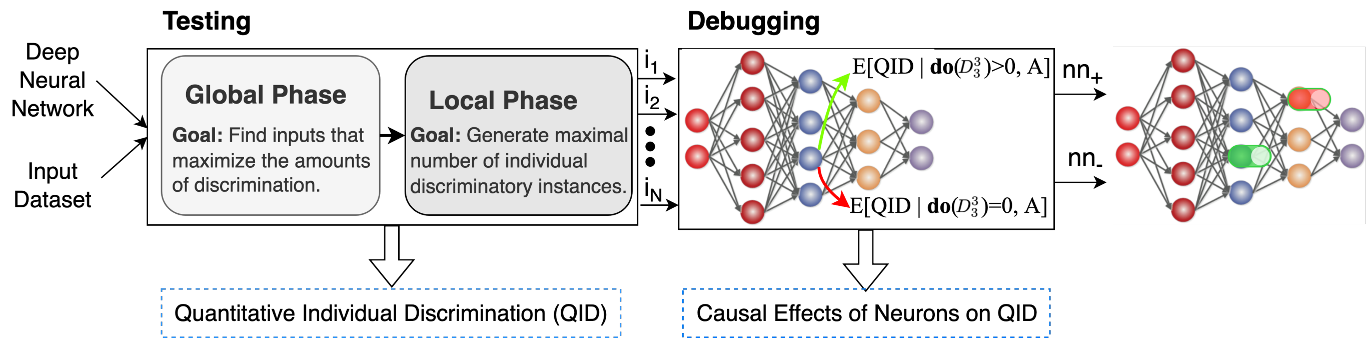

Fig. 1: Workflow of DICE. Given a DNN and relevant input dataset, DICE quantifies QID discrimination via testing and applies causal debugging to localize and mitigate QID discrimination.

returns a set of interesting inputs. Second, DICE uses those inputs to localize neurons with the largest causal effects on the amounts of discrimination. In doing so, it intervenes [30] over a set of suspicious neurons. For every neuron, our debugging approach forces the neurons to be active ($n{>}0$) and non-active ($n{=}0$) over the test cases as far as the functional accuracy of DNN remains in a valid range. Then, it computes the difference between the amounts of QID in these two cases to characterize the causal effects of the neuron on fairness.

DICE reports top k neurons that both have positive impacts (i.e., their activation reduces the amounts of discrimination) and negative impacts (i.e, their activation increases the amounts of discrimination). A potential mitigation strategy is to intervene to keep a small set of neurons activated (for the positive neurons) or deactivated (for the negative neurons).

Test Cases. Consider the *adult census income* [31] dataset with a pre-trained model with 6 layers [17] to overview DICE in practice. We ran DICE for 1 hours and obtain $230,593$ test cases. It discovered 36 clusters from the initial of 14 clusters, and the amounts of QID are 4.05 and 2.64 bits for min entropy and Shannon entropy, respectively, out of a total of 5.3 bits of information from the protected attributes. Considering to order the test cases, we have 6 test case with maximum QID discrimination of 5.3 bits. In addition, we have 29 and 112 test cases cases with 5.2 and 5.1 bits of QID discrimination. The reported numbers are averaged (and rounded) for 10 runs.

Localization and Mitigation. DICE uses the generated test cases to localize layers and neurons with a significant causal contribution to the discrimination. For the census dataset, it identifies the second layer as the layer with largest sensitivity to protected attributes. Among the neurons in this layer, DICE found that 15th neuron has the largest negative influence on fairness (the discrimination decreased by 19.6% when it is deactivated) and 19the neuron has the largest positive influence on fairness (the discrimination decreased by 17.6% when it is activated). Following this localization, a simple mitigation strategy of activating or deactivating these neuron reduces the amounts of QID discrimination by 20% with 3% accuracy loss.

Comparison to the State-of-the-art. We compare DICE to the state-of-the-art techniques in terms of generating individual discrimination (ID) (rather than the quantitative notion) per each protected attribute. Our goal is to evaluate whether the clustering-based search is effective in generating discriminatory instances. We run DICE and baseline for 15 minutes, and report average of results over 10 runs. The baseline includes AEQUITAS [19], ADF [17], and NEURONFAIR [18]. Considering sex as the protected attribute in the census dataset, DICE generated $79.0k$ instances whereas AEQUITAS, ADF, and NEURONFAIR generated $10.4k$, $18.2k$, and $21.6k$ discriminatory instances, respectively. Overall, DICE generate more ID instances in all cases with more success rates. However, DICE is slower in finding the first ID instance in order of a few seconds (in average), since our approach does not generate ID instances in global phase. When considering the time to the first 1,000 instances, DICE has significantly outperformed the state-of-the-art. We conjecture that the improvements are due to smooth search space via quantitative feedback.

IV. PROBLEM STATEMENT

We consider DNN-based classifiers with the set of input variables A partitioned into protected set of variables Z (such as race, sex, and age) and non-protected variables X (such as profession, income, and education). We further assume the output to consist of t prediction classes.

Definition IV.1 (DNN: Semantics)**.** *A deep neural network (DNN) encodes a function $\mathcal{D} : X \times Z \to [0,1]^t$ where $X = X_1 \times X_2 \cdots \times X_n$ is the set of non-protected input variables, $Z = Z_1 \times Z_2 \cdots \times Z_r$ is the set of protected input variables, and the output is t-dimensional probabilistic vector corresponding to t prediction classes. The predicted label $\mathcal{D}_\ell(x,z)$ of an input pair (x,z) is the index of the maximum score, i.e. $\mathcal{D}_\ell(x,z) = \max_i \mathcal{D}_\ell(x,z)(i)$. We assume that the set of protect input variables are finite domain, and we let m be the cardinality of the set of protected variables Z.*

Definition IV.2 (DNN: Syntax)**.** *A DNN $\mathcal{D}$ is parameterized by the input dimension $n{+}r$, the output dimension t, the depth of hidden layers N, and the weights of its hidden layers $W_1, W_2, \ldots, W_N$. Our goal is to test and debug a pre-trained neural network with known parameters and weights. Let D_i*

be the output of layer i that implements an affine mapping from the output of previous layer D_{i-1} and its weights W_{i-1} for $1 \leq i \leq N$ followed by

1) *a fixed non-linear activation unit (e.g., ReLU defined as $D_{i-1} \mapsto \max\{W_{i-1}.D_{i-1}, 0\}$) for $1 \leq i < N$, or*
2) *a SoftMax function that maps scores to probabilities of each class for $i = N$.*

Let D_i^j be the output of neuron j at layer i.

Individual Discrimination. We say a DNN $\mathcal{D}$ is biased based on causal discrimination notion [16], [11], [17], [18] if

$$\exists z_1, z_2 \in Z, x \in X \ s.t. \ \mathcal{D}_\ell(x, z_1) \neq \mathcal{D}_\ell(x, z_2),$$

for $z_1 \neq z_2$ of protected inputs. Intuitively, the idea is to find an individual such that their counterfactual with different protected attributes such as race receives a different outcome.

Quantitative Individual Discrimination. In the setting of fairness testing, it is often desirable to quantify the *amounts* of bias for individuals. We define the notion quantitative individual discrimination (QID) based on the equivalence classes induced from the output of DNN over protected attributes. Formally, $QID(Z, X = x) = \langle Z_1, \ldots, Z_k \rangle$ that is the quotient space of Z characterized by the DNN outputs under an individual with non-protected value x. Using this notion, we say a pair of protected values z, z' are in the same equivalence class i (i.e., $z, z' \in Z_i$) if and only if $\mathcal{D}(z, x) \approx \mathcal{D}(z', x)$.

Given that Z is uniformly distributed and $\mathcal{D}$ is a deterministic function, we can quantify the QID notion for an individual (z, x) according to the Shannon and min entropy, respectively:

$$Q_1(Z,x) = \log_2(m) - \sum_{i=1}^{k} \frac{|QID_i(Z,x)|}{m} . \log_2(|QID_i(Z,x)|)$$

$$Q_\infty(Z, x) = \log_2(m) - \log_2(\frac{m}{k}) = \log_2(k).$$

where m is the cardinality of Z, $|QID_i(Z, x)|$ is the size of equivalence class i, and k is the number of equivalence classes.

Debugging/Mitigating DNN for QID. After characterizing the amounts of discrimination via QID, our next step is to localize a set of layers and neurons that *causally* effect the output of DNN to have k equivalence classes.

Causal logic [30] provides a firm foundation to reason about the causal relationships between variables. We consider a structural causal model (SCM) with exogenous variables U over the unobserved input factors, endogenous variables V over (X, Z, D_i^j); and the set of functions $\mathcal{F}$ over the set V using the DNN function $\mathcal{D}$ and exogenous variables U. Using the SCM, we aim to estimate the average causal effect (ACE) [30] of neuron D_i^j on the QID.

A primary tool for performing such computation is called $\mathtt{do}$ logic [20]. We write $\mathtt{do}(i, j, y)$ to indicate that the output of neuron j at layer i is intervened to stay y. In doing so, we remove the incoming edges to the neuron and force the output of neuron to take a pre-defined value y, but we are not required to control back-door variables due to the feed-forward structure of DNN. Then, the ACE of neuron D_i^j on the quantitative individual discrimination with min entropy can be written as $E[Q_\infty | \mathtt{do}(i, j, y), k, l]$, which is the expected QID after intervening on the neuron given that the non-intervened DNN characterized k classes with an accuracy of l. Our goal is to find neurons with the largest causal effects on the QIDs, requiring that such interventions are faithful to the functionality of DNN.

Definition IV.3 (Quantitative Fairness Testing and Debugging)**.** *Given a deep neural network model $\mathcal{D}$ trained over a dataset A with protected ($Z \subset A$) and non-protected ($X \subset A$) attributes; the search problem is to find a single non-protected value $x \in X$ such that the quantitative individual discrimination (QID), for a chosen measure Q_1 or Q_∞, is maximized over the m protected values $\Sigma = \{z_1, \ldots, z_m\}$. Given the inputs (Σ, x) characterizing the maximum QID, our debugging problem is to find a minimal subset of layers $l \subset \{1, \ldots, N\}$ and neurons $D_l^{(J)}$ for $J \subseteq |W_l|$ such that the average causal effects of D_l^J on the QID are maximum.*

V. Approach

Characterizing Quantitative Individual Discrimination. Given a DNN $\mathcal{D}$ over a dataset A, our goal is to characterize the worst-case QID over all possible individuals. Since min entropy characterizes the amounts of discrimination from one prediction with $Q_\infty(Z, x) \geq Q_1(Z, x)$, it is a useful notion to prioritize the test cases. Therefore, we focus on Q_∞ and propose the following objective function:

$$\max_{x \in X} \ 2^{Q_\infty(Z,x)} + (1 - exp(-0.1 * \delta))$$

where $2^{Q_\infty(Z,x)} = k$ and δ is the maximum distance between equivalence classes, normalized with the exponential function to remain between 0 and 1. The term is used to break ties when two instances characterize the same number of classes, by preferring one with the highest distance. Overall, the goal is to find a single value of non-protected attribute x such that the neural network model $\mathcal{D}$ predicts many distinguishable classes of outcomes when x is paired with m protected values. However, finding those inputs requires an exhaustive search in the exponential set of subsets of input space, and hence is clearly intractable. We propose a gradient-guided search algorithm that aims to search the space of input variables (attributes) to maximize the number of equivalence classes and generate as many discrimination instances as possible.

Search Approach. Our search strategy consists of global and local phases as in some of the prior work [17], [29], [18]. The goal of the global phase is to find the maximum quantitative individual discrimination via gradient-guided clustering. The local phase uses the promising instances to generate a maximum number of discriminatory instances (ID).

Global Phase. Given a current instance x, the global stage first uses m different values from the space of protected attributes, while keeping the values of non-protected attributes the same. Then, it receives m prediction scores from the DNN and partitions them into k classes. We adapt a constrained-based clustering with ϵ where two elements cannot be in the same cluster if their scores differ more than ϵ. Now, the critical step is to perturb the current instance over a subset of non-protected attributes with a direction that will likely increase the number of clusters induced from the perturbed instance in the next step of global search.

In doing so, we first compute the gradients of DNN loss function for a pair of instances (say a, a') in the cluster with the maximum elements. The intuition is that we are more likely to split the largest cluster into 2 or more sub-clusters and increase the number of partitions in the next step. For the pair of samples, we use the non-protected attributes that have the same direction of gradients d since it shows the high sensitivity of loss function with respect to small changes on those common features of the pair. If we were to use gradients of opposite directions, we will neutralize the effects of gradients since we only perturb one instance over the non-protected attributes. Finally, we perturb the current sample x to generate x' using the direction d and step size s_g.

Local Phase. Once we detect an instance with more than 2 clusters, we enter a local phase where the goal is to generate as many discriminatory instances (ID) as possible. In our quantitative approach, we say that an unfavorable decision for an individual x is discriminatory if there is a counterfactual individual x' that received a favorable outcome. Similar to the state-of-the-art [19], [17], [18], we use non-linear optimizer that takes an initial instance x, a step function to generate the next instance around the neighborhood of the current instance, and an objective function that quantifies the discrimination of the current instance. Since our approach uses a continuous objective based on the characteristics of clusters, it enables us to guide the local search to generate discriminatory instances.

Algorithm 1: DICE (SEARCH)

Input: Dataset A, deep learning model $\mathcal{D}$, the loss function for the DNN J, protected attributes P, non-protected attributes NP, the number of partitions over the dataset p, the step size in global perturbation s_g, the step size in local perturbation s_l, the maximum number of global iterations N_g, the maximum number of local iterations N_l, the tolerance ϵ, and time-out T.

Output: Num. Clusters and (local+Global) Test Cases.

```
1  A', cur ← KMeans(A, p), time()
2  while time() - cur < T do
3      x, i, k, δ ← pick(A'), 0, 1, 0.0
4      while i < N_g do
5          I_m, S_m ← Generate_Predict(a, P)
6          X_k, δ' ← Clust(S_m, ϵ)
7          a, a' ← Choose_Pair_Max(X_k)
8          Gs ← (∇J(a), ∇J(a'))
9          d ← choose_common_direct(Gs, NP)
10         x' ← perturb(x, d, s_g)
11         if (|X_k| > k) or (|X_k| = k and δ' > δ) then
12             eval_f(x) {
13             I'_m, S'_m ← Generate_Predict(x, P)
14             X_k' ← Clust(S'_m, ϵ)
15             Δ ← arg.max(X_k') − arg.min(X_k')
16             local_inps.add(x)
17             Return −Δ}
18             step_f ← λ_x perturb_local(x, s_l)
19             LBFGS(x, eval_f, step_f, N_l)
20         global_inps.add(a)
21         k, x ← max(k, |X_k|), x'
22 return k, I = global_inps ∪ local_inps
```

Search Procedure. Algorithm 1 sketches our search algorithm to quantify the amounts of bias. We first use the clustering (KMeans algorithm) to partition the data points into p groups (line 1). Next, we run the algorithm until the time-out T reaches where in each iteration we seed a sample randomly from one of the partitions p (line 2). Then, we proceed into global and local phases of search.

In the global phase, we first use the seed instance x and perturb the protected attributes P to generate m possible instances X_m with different protected values and the same non-protected ones. Then, we run X_m through the DNN model to get the probability scores S_m (line 5). Then, we cluster the scores S_m into k groups using the tolerance ϵ (line 6). Afterward, we compute the gradients over two random instances (from the cluster with the largest size) and use a subset of non-protected features that have the largest number of agreements on the gradient directions. Then, we use those features and their directions to perturb the inputs x and generate the next instance x' (line 7-10). Then, we enter the local phase if the number of clusters or distance between are increased (line 11).

In the local phase, we use the general-purpose optimizer, known as `LBFGS` [32], which takes an initial seed x, an objective function, a step function, and the maximum number of local iterations N_l; it returns the generated instances during the optimization (Line 12-19). In the objective function shown with `eval_f` (Line 12-17), we generate m instances with the same non-protected values but different protected ones (Line 13). We generate prediction scores for those instances and cluster them with tolerance parameter ϵ (line 14). Then, we compute the difference between the indices of two clusters with the smallest and largest scores (line 15). Finally, we record the generated sample and return the difference as the evaluation of optimizer at the current sample (line 16-17). The step function is shown with `perturb_local` (Line 18) where it guides the optimizer to take one step in the input space. Our step function uses a random sample from a different cluster compared to the current sample. Then, it computes the normalized sum of gradients and perturbs it using the smallest gradients to remain in the neighborhood of the current sample.

Debugging Approach. Since it is computationally difficult to intervene over all possible neurons in a DNN, we first adapt a layer-localization technique from the literature of DL framework debugging [33], [21] where we detect a layer with the largest sensitivity to the protected attributes. Let $D_i(z, x)$ be the output of layer i over protected value z and non-protected value x. Let $\Delta_i(x) : \mathbb{R}^{|D_i|} \times \mathbb{R}^{|D_i|} \to \mathbb{R}$ be the distance between the outputs of DNN at layer i as triggered by m different protected values and the same non-protected value x, and let δ_i be the $\max_x \Delta_i(x)$. The rate of changes in the sensitivity of layer i (w.r.t protected attributes) is

$$\rho_i = \frac{\delta_i - \max_j \delta_j}{\max_j \delta_j + \epsilon},\ with\ 0 \leq\ j <\ i$$

where $\delta_0 = 0.0$ and $\epsilon = 10^{-7}$ (to avoid division-by-zero [33], [21]). Let $l = \arg\max_i \rho_i$ be the layer index with the maximum rate of changes. Our next step is to localize neurons in the layer l that have significant positive or negative effects on fairness. Let V_l^j be the set of possible values for neuron j at later l (recorded during the layer localization). We are interested in computing the average causal effects when the neuron D_l^j is activated vs. deactivated, noting that such interventions might affect the functionality of DNN. Therefore, among a set of intervention values, we choose one activated value $v_1 \in V_l^j > 0$ and one deactivated value $v_2 \in V_l^j \approx 0$, considering the functional accuracy of DNN within ϵ of original accuracy A. Therefore, we define average causal difference (ACD) for a neuron D_l^j as:

$$\mathbb{E}[Q_\infty \mid \mathtt{do}(l, j, v_1 > 0), k, A] - \mathbb{E}[Q_\infty \mid \mathtt{do}(l, j, v_2 \approx 0), k, A],$$

where do notation is used to force the output of neuron j at layer l to a fix value v. We then return the neuron indices with the largest positive (aggravating discrimination) and smallest negative (mitigating discrimination). Let $\hat{i}$ and $\hat{j}$ be the layer and neuron with the largest positive ACD. One simple mitigation strategy is thus to deactivate the neuron $D_{\hat{i}}^{\hat{j}}$, expecting to reduce QID by ACD/k percentage. Similarly, activating the neuron with the smallest negative ACD is expected to reduce QID by ACD/k percentage.

Debugging Procedure. Algorithm 2 shows the debugging aspect of DICE. Given a set of test cases from the search algorithm, we first use a notion of distance (e.g, $\Delta = L_1$) to compute the difference between any pair of protected values z, z' w.r.t the outputs of layer $l \in \{1, \ldots, N\}$ (line 1). Then, we compute the rate of changes (line 2-3) and return a layer l with the largest change (line 4). We compute various statistics on the output of every neuron i at layer l such as the minimum, maximum, average, average±std. dev, average±2∗std. dev, etc (line 5). Among those values, we take the smallest and largest values such that the intervention on the neuron i at layer l has the minimal impacts on the accuracy of DNN (line 6). Finally, we compute the average causal difference (line 7) and return the indices of layer, neurons with large negative influence, and neurons with large positive influence.

VI. EXPERIMENTS

Datasets and DNN models. We consider 10 socially critical datasets from the literature of algorithmic fairness. These datasets and their properties are described in Table I. For the DNN model, we used the same architecture as the literature [17], [18], [29] and trained all datasets on a six-layers fully-connected neural network with $\langle 64, 32, 16, 8, 4, 2\rangle$ neurons. We used the same hyperparameters for the all training with num_epochs, batch_size, and learning_rate are set to 1000, 128, and 0.01, respectively. The accuracy of trained models are reported in Table V.

Algorithm 2: DICE (DEBUGGING).

Input: Dataset $A = (A_X, A_Z)$, $\mathcal{D}$ with accuracy $\mathcal{A}_\mathcal{D}$, Test cases I, the distance function Δ, the tolerance of layer localization ϵ_1, the tolerance of accuracy loss ϵ_2, and k top items.

Output: Layer Index, Negative, and Positive Neurons.

`/* Layer Localization */`

1 $\delta \leftarrow \lambda_l\ \max \sum_{x \in I} \Delta\big(D_l(z, x), D_l(z', x)\big)$

2 $\delta[0]$, $\delta_{max} \leftarrow 0.0$, $\lambda_i \max_{j<i} \delta[j]$

3 $\rho \leftarrow \lambda_l \frac{\delta[i] - \delta_{max}[i]}{\delta_{max}[i] + \epsilon_1}$

4 $l \leftarrow \arg\max_i \rho[i]$

`/* Neuron Localization */`

5 $V_l \leftarrow \lambda_i\ \mathtt{stats}(D_l^i)$

6 $v_l \leftarrow \lambda_i\ \lambda_j\ |\mathcal{A}_\mathcal{D} - \mathcal{A}_{\mathcal{D} \leftarrow do(l, i, V_l[j])}| \leq \epsilon_2$

7 $ACD_l \leftarrow \lambda_j\ \mathbb{E}\ (k' | \mathrm{do}(l, j, v_l^j > 0))$ - $\mathbb{E}(k' | \mathrm{do}(l, j, v_l^j = 0))$

8 **return** l, $top_k(\max ACD_l)$, $top_k(\min ACD_l)$.

Technical Details. We implemented DICE with TensorFlow v2.7.0 and scikit-learn v0.22.2. We run all the experiments on an Ubuntu 20.04.4 LTS OS sever with AMD Ryzen Threadripper PRO 3955WX 3.9GHz 16-cores X 32 CPU and two NVIDIA GeForce RTX 3090 GPUs. We choose the values 10, 1000, 0.025, 1, and 1 for max_global, max_local, ϵ, s_g, and s_l in Algorithm 1, respectively, and take the average of 10 multiple runs for all experiments. In Algorithm 2, we used L_1-norm, 10^{-7}, 0.05, and 3 for Δ, ϵ_1, ϵ_2, and k, respectively.

Research Questions. We seek to answer the following three questions using our experimental setup.

- **RQ1** Can DICE characterize the amounts of information from protected attributes used for the inferences?
- **RQ2** Is the the proposed search algorithm effective and efficient (vis-a-vis the state-of-the-art techniques) in generating individual discrimination instances?
- **RQ3** Can the proposed causal debugging guide us to localize and mitigate the amounts of discrimination?

> Our open-source tool DICE with all experimental subjects are publicly accessible:
> - https://github.com/armanunix/Fairness-testing

A. Characterizing QID via Search (RQ1)

An important goal is to characterize the amount of information from protected attributes used during the inference of DNN models. Table II shows the result of experiments to answer this research question. The left side of table shows the

TABLE I: Datasets used in our experiments.

Dataset	#Instances	#Features	Protected Groups Name	Size	Num. Protected Values (m)	Outcome Label Label 1	Label 0
Adult *Census* Income [31]	32, 561	13	Sex	2	90	High Income	Low Income
			Race	5			
			Age	9			
Compas [34]	7, 214	12	Sex	2	12	Did not Reoffend	Reoffend
			Race	2			
			Age	3			
German Credit [35]	600	20	Sex	2	16	Good Credit	Bad Credit
			Age	8			
Default Credit [36]	13, 636	23	Sex	2	12	Default	Not Default
			Age	6			
Heart Health [37]	297	13	Sex	2	14	Disease	Not Disease
			Age	7			
Bank Marketing [38]	45, 211	16	Age	9	9	Subscriber	Non-subscriber
Diabetes [39]	768	8	Age	9	9	Positive	Negative
Students Performance [40]	1044	32	Sex	2	16	Pass	Not Pass
			Age	8			
MEPS15 [41]	15, 830	137	Age	9	36	Utilized Benefits	Not Utilized Benefits
			Race	2			
			Sex	2			
MEPS16 [41]	15, 675	137	Age	9	36	Utilized Benefits	Not Utilized Benefits
			Race	2			
			Sex	2			

initial characteristics such as the number of protected values (m), the maximum possible amounts of discrimination (Q_I) based on $\mathtt{min}(\epsilon^{-1}, m)$, and the initial number of clusters found using samples from the dataset (K_I). The right side of table shows the results after running our search for 1 hour. The column #I is the number of QID instances generated, and K_F is the maximum number of clusters discovered by DICE. The column T_{K_F} is the time taken to find the maximum number of clusters from an input with initial clusters K_I (in seconds). The columns Q_∞ and Q_1 are the quantitative individual discrimination based on min entropy and Shanon entropy, respectively. The columns #$I_{K_F^1}$, #$I_{K_F^2}$, and #$I_{K_F^3}$ show the number of test cases with the highest, second-highest, and third-highest QIDs, respectively, that order test cases with their QID severity. Overall, the results show that DICE can find $3.4\times$ more clusters (in average) from the initial characteristics within one minute of search. The DNN for Students dataset showed the largest increase in the number of clusters going from 1.9 to 10.9. DICE found that Adult Income Census dataset has the largest amounts of QID where 4.05 out of 5.3 bits (76.4%) from protected variables are used to make decisions. The German Credit dataset with 1.61 out of 4.0 bits (40.0%) showed the least amounts of discrimination. For test-case prioritizing, the column #$I_{K_F^1}$ shows our approach to be useful in finding a small percentage of generated test cases with the worst-case discrimination. In 7 out of 10 experiments, DICE found less than 50 test cases with severe discrimination out of hundreds of thousands inputs.

Answer RQ1: The search algorithm is effective in characterizing the amounts of discrimination via QID. Within 1 hour, it increased the number of clusters by $3.4\times$ in average, and found instances that used up to 76% of protected information (4.05 out of 5.3 bits) to infer DNN outcomes. DICE is useful to prioritize test cases with their severity where it generates less than 50 test cases with the maximum QID among hundreds of thousands test cases.

B. Individual Discriminatory Instances (RQ2)

In this section, we compare the efficiency and effectiveness of our search algorithm to the state-of-the-art techniques in searching individual discrimination (ID) instances (as defined in Section IV). Our baselines are AEQUITAS [19], ADF [17], and NEURONFAIR [18]. We obtained the implementations of these tools from their GitHub repositories and configured them according to the prescribed setting to have the best performance. Following these techniques, we report the results for each protected attribute separately. Table III shows the results of baselines and DICE in runs of 15 minutes. The results are averaged over 10 repeated runs. The column #ID is the total number of generated individual discriminatory instances. The column l_s is the success rate of local stage of searches. We exclude the global success rate since the goal of global phase in our search is to maximize QID whereas the local phase focuses to generate many ID instances. We calculate success rate as the number of ID found over the total number of generated samples. The columns $T.1st$ and $T.1k$ are the amount of time (in seconds) taken to find the first ID instance and to generate $1,000$ individual discriminatory instances (note: N/A in column $T.1k$ means that the tool did not generate $1,000$ IDs in the experiment timeout of 900 seconds in the average of 10 runs).

The result shows that DICE outperforms the-state-of-the-art in generating many ID instances. In particular, DICE finds $27.1\times$, $16.0\times$, and $16.0\times$ more IDs in the best case compare to AEQUITAS, ADF, and NEURONFAIR, respectively. DICE also generates $3.2\times$, $2.3\times$, and $2.6\times$ more IDs in the worst case compare to AEQUITAS, ADF, and NEURONFAIR, respectively. The success rate of local search are 20.6%, 33.0%, 29.6%, and 78.2% in average for AEQUITAS, ADF, NEURONFAIR, and DICE, respectively. For the time taken to find the first ID, AEQUITAS achieves the best result with an average of 0.03 (s) whereas it took DICE 1.46 (s) in average to find the first ID. In average, DICE was found to take the lowest time to generate 1000 IDs with 57.2 (S), while ADF

TABLE II: DICE characterizes QID for 10 datasets and DNNs in 1 hour run (results are the average of 10 runs).

Dataset	m	Q_I	K_I	#I	K_F	T_{K_F}	Q_∞	Q_1	#$I_{K_F^1}$	#$I_{K_F^2}$	#$I_{K_F^3}$
Census	90	5.3	13.54	$230,593$	35.61	21.04	4.05	2.64	6.0	28.6	111.6
Compas	12	3.6	3.12	$157,968$	10.24	6.50	1.81	1.40	35.2	338.7	$1,016.9$
German	16	4.0	2.34	$245,915$	9.56	13.14	1.61	1.10	6.6	16.2	54.8
Default	12	3.6	5.58	$258,105$	11.26	10.94	2.10	1.78	$3,528.8$	$9,847.2$	$9,771.0$
Heart	14	3.8	4.54	$270,029$	10.01	11.88	2.31	1.80	21.7	135.2	579.7
Bank	9	3.2	1.45	$172,686$	8.93	3.68	2.25	1.98	$5,118.5$	$13,513.3$	$20,438$
Diabetes	10	3.3	2.39	$504,414$	7.90	0.016	1.40	1.11	89.7	609.6	$2,310.1$
Students	16	4	1.90	$133,221$	10.90	14	1.93	1.35	16.0	130.7	128.7
MEPS15	36	5.2	7.03	$19,673$	18.52	31.62	2.61	1.62	2.6	3.5	6.0
MEPS16	36	5.2	9.06	$14,266$	19.25	49.16	2.21	1.52	2.0	3.5	6.0

took 179.1 (s), NEURONFAIR took 135.4 (s), and AEQUITAS took the longest time at 197.7 (s). Overall, our experiments indicate that DICE is effective in generating ID instances compared to the three state-of-the-art techniques, largely due to the smoothness of the feedback during the local search.

Answer RQ2: Our experiments demonstrate that DICE outperforms the state-of-the-art fairness testing techniques [19], [17], [18]. In the best case, our approach found $20\times$ more individual discrimination (ID) instances than these techniques with almost $3\times$ more success rates in average. However, we found that DICE is slower than those techniques in finding the first ID instance in the order of a few seconds.

C. Causal Debugging of DNNs for Fairness (RQ3)

We perform experiments over the DNN models to study whether the proposed causal debugging approach is useful in identifying layers and neurons that significantly effect the amounts of discrimination as characterized by QID. Table IV shows the results of experiments (averaged over 10 independent runs). The first two columns show the localized layer and its influence (i.e., l and ρ in Algorithm 2). The next six columns show top 3 neurons with the positive influence on fairness (i.e., activating those neurons reduce the amounts of discrimination based Q_∞). The last six columns show top 3 neurons with the negative influence on fairness (i.e., activating those neurons increase the amounts of discrimination based Q_∞). The layer index 2 is more frequently localized than other layers where the layers 3, 4, and 5 are localized once. Overall, the average causal difference (ACD) ranges from 4% to 55% for neurons with positive fairness effects and from 0.6% to 18.3% for neurons with negative fairness effects.

Guided by localization, DICE intervenes to activate neurons with positive fairness influence or de-activate those with negative influence. Table V shows the results of this mitigation strategy. The columns A and K show the accuracy and the number of clusters (averaged over a set of random test cases) reported by DICE before mitigation over the DNN model. The columns $A^{=0}$ and $K^{=0}$ are accuracy and the number of clusters reported after mitigating the DNN model by *de-activating* the neuron with the highest negative fairness impacts (as suggested by Neuron_1^- in Table IV). Similarly, the columns $A^{>0}$ and $K^{>0}$ are accuracy and the number of clusters reported after mitigating the DNN model by *activating* the neuron with the highest positive fairness impacts (as suggested by Neuron_1^+ in Table IV). The results indicate that the activation interventions can reduce QID discrimination by at least 5% with 3% loss of accuracy and up to 64.3% with 2% loss of accuracy. The de-activation, on the other hand, can improve the fairness by at least 6% with 1% loss of accuracy and up to 27% with 2% loss.

Answer RQ3: The debugging approach implemented in DICE identified neurons that have at least 5% and up to 55% positive causal effects on the fairness and those which have at least 0.6% and up to 18.3% negative causal effects. A mitigation strategy followed by the localization can reduce the amounts of discrimination by at least 6% and up to 64.3% with less than 5% loss of accuracy.

VII. DISCUSSION

Limitation. In this work, we consider all set of protected values and perturb them to generate counterfactual. Various perturbations of protected attributes may yield unrealistic counterfactuals and contribute towards false positives (an over-approximation of discrimination). This limitation can be mitigated by supplying domain-specific constraints (Age$<$YY $\implies$ NOT(married)): we already apply some common-sense constraints (e.g., to ensure valid range of age). In addition, similar to any dynamic testing methods, our approach might miss discriminatory inputs and is prone to false negatives. The probability of missing relevant inputs can be contained under a suitable statistical testing (e.g., Bayes factor). In addition, our debugging approach is similar to pin-pointing suspicious code fragments and is based on causal reasoning of its effect in decision making rather than correlation. But, it is not to furnish explanations or interpretations of black-box DNN functions.

Threat to Validity. To address the internal validity and ensure our finding does not lead to invalid conclusion, we follow established guideline and take average of repeated experiments. To ensure that our results are generalizable and address external validity, we perform our experiments on 10 DNN models taken from the literature of fairness testing. However,

TABLE III: Comparison of generating discriminatory instances to the state-of-the-art in 900 seconds. PROT. is the the protected attributes; #ID is the number of individual discriminatory instances; l_s is the success rate of the local search; $T.1st$ and $T.1k$ are the time taken to find the first and $1,000^{th}$ discriminatory instance, respectively (in seconds). The results are averaged over 10 runs of each tool where we report the deviation from the mean in the parenthesis. The best outcomes are highlighted in **bold** (k is 1,000 and $\epsilon < 0.1$ used for the standard deviation).

Dataset	Prot.	AEQUITAS [19]				ADF [17]				NEURONFAIR [18]				DICE			
		$\#ID$ (k)	l_s (%)	$T.1$ (s)	$T.1k$ (s)	$\#ID$ (k)	l_s (%)	$T.1$ (s)	$T.1k$ (s)	$\#ID$ (k)	l_s (%)	$T.1$ (s)	$T.1k$ (s)	$\#ID$ (k)	l_s (%)	$T.1$ (s)	$T.1k$ (s)
Censes	sex	10.4 (1.1)	10.3 (1.0)	**0.02** (ϵ)	88.5 (13.1)	18.2 (1.0)	18.3 (0.1)	0.5 (ϵ)	52.1 (7.4)	21.6 (1.3)	19.5 (0.8)	0.05 (ϵ)	38.8 (6.5)	**79.0 (3.2)**	**74.0 (1.6)**	0.5 (ϵ)	**9.8 (0.6)**
	age	8.8 (7.2)	29.1 (2.2)	0.02 (ϵ)	113.5 (34.8)	21.6 (1.2)	55.8 (2.0)	0.5 (ϵ)	33.0 (7.3)	21.8 (11)	54.8 (1.8)	**0.01** (ϵ)	48.9 (19.7)	**112 (2.0)**	**93.4 (0.6)**	0.5 (ϵ)	**8.0 (1.0)**
	race	13.2 (9.7)	25.2 (1.5)	0.02 (ϵ)	89.9 (15.9)	24.1 (9.0)	43.3 (1.0)	0.7 (0.2)	40.6 (10.0)	25.9 (1.1)	43.5 (2.8)	**0.01** (ϵ)	37.5 (9.9)	**107 (1.7)**	**88.1 (0.8)**	0.5 (ϵ)	**8.8 (1.3)**
Compas	sex	12.6 (3.3)	13.8 (3.2)	**0.01** (ϵ)	118.2 (16.2)	17.3 (0.7)	17.4 (0.6)	0.02 (ϵ)	49.0 (14.0)	15.1 (1.1)	14.3 (0.9)	0.02 (ϵ)	56.3 (9.1)	**40.2 (1.5)**	**59.0 (1.4)**	0.2 (ϵ)	**27.6 (5.9)**
	age	6.8 (0.9)	10.3 (1.3)	0.02 (ϵ)	150.3 (23.7)	19.0 (0.9)	25.6 (1.2)	**0.01** (ϵ)	47.6 (5.8)	14.0 (0.8)	18.1 (0.9)	0.01 (ϵ)	70.1 (15.1)	**66.2 (3.0)**	**72.1 (1.6)**	0.01 (ϵ)	**19.7 (2.9)**
German	sex	8.2 (0.8)	12.7 (1.2)	**0.01** (ϵ)	120.7 (20.1)	14.3 (0.7)	22.1 (0.8)	0.02 (ϵ)	68.7 (12.0)	12.9 (0.6)	18.3 (0.8)	0.09 (ϵ)	86.4 (14.5)	**62.6 (1.6)**	**80.2 (0.6)**	0.2 (0.2)	**15.0 (2.1)**
	age	8.4 (0.6)	39.8 (2.1)	0.02 (ϵ)	118.9 (22.9)	13.4 (1.1)	57.4 (3.0)	0.01 (ϵ)	60.6 (12.9)	12.9 (0.8)	52.5 (2.2)	**0.01** (ϵ)	56.5 (12.7)	**78.7 (0.8)**	**93.6 (0.2)**	0.05 (ϵ)	**11.1 (0.5)**
Default	sex	4.6 (0.6)	5.3 (0.6)	0.2 (0.2)	200.0 (45.1)	12.8 (0.7)	18.1 (1.1)	**0.01** (ϵ)	78.8 (7.3)	11.4 (0.9)	14.9 (1.3)	0.01 (ϵ)	80.0 (15.4)	**29.4 (1.5)**	**38.8 (1.9)**	2.1 (1.8)	**22.0 (10.5)**
	age	7.8 (2.5)	17.2 (3.2)	0.08 (ϵ)	130.4 (73.8)	13.3 (1.3)	43.4 (3.1)	**0.02** (ϵ)	78.9 (19.8)	9.0 (1.1)	29.9 (3.5)	0.06 (ϵ)	104.8 (14.7)	**50.4 (3.7)**	**68.4 (2.9)**	0.6 (ϵ)	**36.1 (17.1)**
Heart	sex	7.9 (3.2)	9.1 (3.4)	0.02 (ϵ)	171.8 (30.5)	10.6 (1.0)	13.6 (1.2)	**0.01** (ϵ)	94.9 (16.1)	10.2 (0.5)	10.9 (.5)	0.01 (ϵ)	84.4 (23.9)	**84.4 (1.3)**	**83.9 (1.2)**	0.04 (ϵ)	**7.7 (0.4)**
	age	16.7 (1.7)	48.1 (4.0)	0.02 (ϵ)	79.0 (13.2)	25.8 (1.7)	70.0 (2.5)	**0.01** (ϵ)	44.1 (18.6)	28.7 (1.2)	68.3 (1.4)	0.01 (ϵ)	37.9 (16.5)	**92.5 (1.6)**	**95.1 (0.4)**	0.04 (ϵ)	**7.5 (0.6)**
Bank	age	11.0 (1.8)	38.0 (4.1)	0.04 (ϵ)	115.0 (42.3)	7.9 (1.0)	34.2 (4.2)	**0.02** (ϵ)	136 (39.8)	9.2 (1.1)	36.6 (4.1)	0.03 (ϵ)	107.3 (40.2)	**47.4 (5.6)**	**87.3 (1.4)**	0.4 (0.2)	**25.0 (19.5)**
Diabetes	age	15.3 (1.0)	84.5 (1.7)	**0.01** (ϵ)	29.3 (4.7)	46.8 (0.9)	76.4 (0.8)	0.01 (ϵ)	17.8 (4.2)	47.9 (1.7)	76.2 (1.1)	0.01 (ϵ)	18.9 (4.7)	**171 (1.4)**	**94.2 (0.1)**	0.04 (ϵ)	**5.1 (0.2)**
Student	sex	2.4 (0.4)	6.1 (1.0)	0.03 (ϵ)	419.5 (58.9)	4.7 (0.4)	12.2 (1.0)	**0.01** (ϵ)	200.8 (25.5)	4.1 (0.6)	9.4 (1.3)	0.02 (ϵ)	246.5 (43.6)	**30.7 (1.1)**	**72.9 (1.3)**	1.2 (0.7)	**28.8 (4.0)**
	age	1.6 (0.3)	12.8 (2.3)	0.04 (ϵ)	564.8 (0.2k)	2.7 (0.4)	24.1 (2.5)	0.07 (ϵ)	330.9 (54.0)	2.7 (0.4)	20.8 (2.8)	**0.02** (ϵ)	409.7 (91.5)	**43.3 (1.0)**	**89.3 (0.5)**	0.5 (0.3)	**19.6 (1.4)**
MEPS15	sex	0.6 (0.1)	8.0 (1.4)	**0.01** (ϵ)	N/A	1.0 (0.2)	14.7 (2.2)	0.01 (ϵ)	337.0 (0.4k)	0.9 (0.2)	11.7 (2.3)	0.01 (ϵ)	**68.1 (0.2k)**	**6.2 (0.4)**	**67.9 (2.8)**	7.2 (6.4)	174.2 (33.5)
	age	0.4 (0.1)	16.1 (1.8)	0.03 (ϵ)	N/A	0.8 (0.2)	38.3 (6.8)	**0.02** (ϵ)	**92.0 (0.3k)**	0.7 (0.1)	29.1 (3.5)	0.05 (ϵ)	N/A	**6.7 (0.4)**	**82.1 (2.9)**	3.2 (3.4)	136.0 (12.7)
	race	0.7 (0.1)	9.4 (0.8)	**0.02** (ϵ)	N/A	1.4 (0.2)	21.1 (3.4)	0.02 (ϵ)	683.3 (0.1k)	1.0 (0.2)	14.2 (2.1)	0.02 (ϵ)	474.4 (0.4k)	**6.4 (0.4)**	**71.0 (2.2)**	13 (12)	**146.2 (25.8)**
MEPS16	sex	0.6 (0.1)	7.6 (1.2)	0.02 (ϵ)	N/A	1.2 (0.1)	17.6 (1.9)	**0.01** (ϵ)	605.2 (0.3k)	1.0 (0.1)	12.9 (1.8)	0.01 (ϵ)	256.0 (0.4k)	**5.8 (0.4)**	**74.1 (1.5)**	0.14 (ϵ)	**165.3 (19.3)**
	age	0.3 (ϵ)	13.9 (2.0)	0.02 (ϵ)	N/A	1.0 (0.2)	42.6 (7.9)	1.1 (1.1)	73.8 (0.2k)	1.0 (0.2)	38.1 (5.5)	0.03 (ϵ)	**65.5 (0.2k)**	**6.1 (0.3)**	**79.3 (1.6)**	1.4 (ϵ)	163.7 (12.9)
	race	1.1 (0.2)	14.5 (2.1)	0.01 (ϵ)	654.4 (0.3k)	1.7 (0.2)	26.3 (3.8)	**0.01** (ϵ)	509.3 (0.1k)	2.1 (0.3)	27.1 (2.7)	0.01 (ϵ)	480.4 (49.1)	**6.3 (0.4)**	**76.9 (2.7)**	0.2 (ϵ)	**163.6 (20.2)**

TABLE IV: Localization of layers and neurons that causally influence fairness. Neuron_i^+ shows the index of i-th top neuron that has positive influence on fairness once activated; Neuron_i^- shows the index of i-th top neuron that has negative influence on fairness once activated; ACD_i^+ shows the (normalized) average causal difference of i-th top neuron with positive fairness influence; ACD_i^- shows the (normalized) average causal difference of i-th top neuron with negative fairness influence.

Dataset	Layer Index	Layer Influence	Neuron_1^+	ACD_1^+	Neuron_2^+	ACD_2^+	Neuron_3^+	ACD_3^+	Neuron_1^-	ACD_1^-	Neuron_2^-	ACD_2^-	Neuron_3^-	ACD_3^-
Census	2	9.01	N_{19}	0.159	N_3	0.150	N_{12}	0.122	N_{15}	0.183	N_{24}	0.163	N_{14}	0.102
Compas	2	20.70	N_{25}	0.516	N_{28}	0.419	N_7	0.346	N/A	N/A	N/A	N/A	N/A	N/A
German	5	1.79	N_3	0.050	N_1	0.013	N/A	N/A	N_0	0.042	N/A	N/A	N/A	N/A
Default	2	27.58	N_5	0.039	N_{27}	0.022	N_{19}	0.014	N_2	0.031	N_{13}	0.027	N_{10}	0.019
Heart	4	1.47	N_5	0.120	N_4	0.048	N_0	0.044	N/A	N/A	N/A	N/A	N/A	N/A
Bank	3	6.62	N_0	0.495	N_7	0.178	N_1	0.091	N_6	0.057	N_{11}	0.014	N/A	N/A
Diabetes	2	1.67	N_{19}	0.041	N_6	0.035	N_5	0.031	N_0	0.042	N_{29}	0.001	N/A	N/A
Students	2	4.01	N_{22}	0.550	N_{24}	0.442	N_8	0.229	N_4	0.084	N_{18}	0.055	N_{28}	0.026
MEPS15	2	35.44	N_{24}	0.230	N_6	0.167	N_{14}	0.160	N/A	N/A	N/A	N/A	N/A	N/A
MEPS16	2	47.46	N_8	0.147	N_{11}	0.138	N_{24}	0.114	N_{30}	0.006	N_{30}	0.003	N_{22}	0.001

TABLE V: A is accuracy, K is the average number of clusters from test cases; $A^{=0}$ is the accuracy after deactivating the neuron with the highest negative fairness impacts; $K^{=0}$ is the average number of clusters after the deactivation; $A^{>0}$ is the accuracy after activation the neuron with the highest positive fairness impacts; $K^{>0}$ is the average number of clusters after the activation; and T_I is the amount of computation times for localization and mitigation in seconds.

Dataset	A	K	$A^{=0}$	$K^{=0}$	$A^{>0}$	$K^{>0}$	T_I
Census	0.882	22.68	0.85	18.22	0.85	18.68	$1,538$
Compas	0.976	5.19	N/A	N/A	0.971	2.66	$1,226$
German	1.0	5.05	0.992	4.62	0.956	3.53	559
Default	0.827	8.08	0.822	6.70	0.826	7.70	$1,230$
Heart	0.96	6.91	N/A	N/A	0.916	6.59	553
Bank	0.923	5.95	0.901	4.32	0.893	3.16	858
Diabetes	0.993	3.90	0.956	2.91	0.965	3.61	$1,238$
Students	1.0	6.42	0.97	5.79	0.98	2.29	$1,206$
MEPS15	0.898	9.35	N/A	N/A	0.866	6.93	$1,348$
MEPS16	0.913	7.42	0.906	6.98	0.903	6.27	$1,368$

it is an open problem whether these datasets and DNN models are sufficiently representative for fairness testing.

VIII. RELATED WORK

Fairness Testing of ML systems. THEMIS [10] presents a causal discrimination notion where they measure the difference between the fairness metric of two subgroups by *counterfactual* queries; i.e., they sample individuals with the protected attributes set to A and compare the outcome to a counterfactual scenario where the protected attributes are set to B. Symbolic generation (SC) [28], [22] presents a black-box testing that approximates the ML models with decision trees and leverage symbolic execution over the tree structure to find individual discrimination (ID). AEQUITAS [19] uses a two-step approach that first uniformly at random samples instances from the input dataset to find a discriminatory instance and then locally perturb those instances to further generate biased test cases. EXPGA [16] proposed a genetic algorithm (GA) to generate ID instances in natural language processes. The proposed technique used a prior knowledge graph to guide the perturbation of protected attributes in the NLP tasks. While these techniques are black-box, they potentially suffer from the lack of local guidance during the search. ADF [17] utilized the gradient of the loss function as guidance in generating ID instances. The global phase explores the input space to find diverse set of individual discrimination whereas the local phase exploits each instance to generate many individual discriminatory (ID) instances in their neighborhoods. EIDIG [29] follows similar ideas to ADF, but uses different computations of gradients. First, it uses the gradients of output (rather than loss function) to reduce the computation cost at each iteration. Second, it uses momentum of gradients in global phase to avoid local optima. NEURONFAIR [18] extends ADF and EIDIG to support unstructured data (e.g., image, text, speech, etc.) where the protected attributes might not be well-defined. In addition, NEURONFAIR is guided by the DNN's internal neuron states (e.g., the pattern of activation and deactivation) and their activation difference. Beyond the capability of these techniques, DICE quantifies the amounts of discrimination, enables software developers to prioritize test cases, and searches multiple protected attributes at one time.

Beyond the scope of this paper, a body of prior work [42], [43], [44], [23], [45], [46] considered testing for group fairness. FAIRWAY [43] mitigates biases after finding suitable ML algorithm configurations. In doing so, they used a multi-objective optimization (FLASH) [47]. PARFAIT-ML [45] searches the hyperparameter space of classic ML algorithms via a gray-box evolutionary algorithm to characterize the magnitude of biases from the hyperparameter configuration.

Debugging of Deep Neural Network. CRADLE [33] traced the execution graph of a DNN model over two different deep-learning frameworks and used the differences in the outcomes to localize what backend functions might cause a bug. However, since CRADLE did not use causal analysis, it showed a high rate of false positive. AUDEE [48] used a similar approach, but it leveraged causal-testing methods. In particular, it designed strategies to intervene in the DNN models and tracked how the intervention affected the observed inconsistencies. We adapted the layer localization of CRADLE and AUDEE; but our causal localization is developed using `do` logic for a meta-property (fairness). AUDEE used a simple perturbation of neuron values for functional correctness (i.e., any inconsistency shows a bug) without considering the accuracy or the severity of neuron contributions to a bug.

In-process Mitigation. A set of work considers in-process algorithms to mitigate biases in ML predictions [49], [50], [51]. Adversarial debiasing [49] and Prejudice remover [50] improve fairness by adding constraints to model parameters or the loss function. Exponentiated gradient [51] uses a meta-learning algorithm to infer a family of classifiers that maximizes accuracy and fairness. Different than these approaches, we develop a mitigation approach that is specialized to handle neural networks for individual fairness. This setting allows us to exploit the layer-based structure of NNs toward causal reasoning and mitigation. We believe that our approach can be extended with in-process mitigation techniques to maximize fairness in the DNN-based decision support systems.

Formal Methods. We believe that this paper can connect to the rich literature of formal verification and its application. Here, we provide two examples. FAIRSQUARE [52] certifies a fair decision-making process in probabilistic programs using a novel verification technique called the weighted-volume-computation algorithm. SFTREE [53] formulated the problem of inferring fair decision tree as a mixed integer linear programming and apply constraint solvers iteratively to find solutions.

Fairness in income, wealth, and taxation. We develop a fairness testing and debugging approach that is uniquely geared toward handling regression problems. Therefore, our approach can be useful to study and address biases in income and wealth distributions [54] among different race and gender. Furthermore, our approach can be useful to study fairness in taxation (e.g., vertical and horizontal equities [55], [56]). We left further study in these directions to future work.

IX. CONCLUSION

DNN-based software solutions are increasingly being used in socio-critical applications where a bug in their design may lead to discriminatory behavior. In this paper, we presented DICE: an information-theoretic model to characterize the amounts of protected information used in DNN-based decision making. Our experiments showed that the search and debugging algorithms, based on the quantitative landscape, are effective in discovering and localizing fairness defects.

Acknowledgement. The authors thank the anonymous ICSE reviewers for their time and invaluable feedback to improve this paper. This research was partially supported by NSF under grant DGE-2043250 and UTEP College of Engineering under startup package.

REFERENCES

[1] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016, http://www.deeplearningbook.org.
[2] S. Ranchordás and L. Scarcella, "Automated government for vulnerable citizens: Intermediating rights," *SSRN Electronic Journal*, 2021.
[3] N. Tishby, F. C. Pereira, and W. Bialek, "The information bottleneck method," *arXiv preprint physics/0004057*, 2000.
[4] S. M. Julia Angwin, Jeff Larson and L. Kirchne, "Machine bias," https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing, 2021, online.
[5] D. A. Elyounes, "" computer says no!": The impact of automation on the discretionary power of public officers," *Vand. J. Ent. & Tech. L.*, vol. 23, p. 451, 2020.
[6] N. Escher and N. Banovic, "Exposing error in poverty management technology: A method for auditing government benefits screening tools," *Proc. ACM Hum. Comput. Interact.*, vol. 4, no. CSCW, pp. 064:1–064:20, 2020. [Online]. Available: https://doi.org/10.1145/3392874
[7] J. Slemrod, "Group equity and implicit discrimination in tax systems," *National Tax Journal*, vol. 75, no. 1, pp. 201–224, 2022.
[8] D. A. Brown, "The IRS is targeting the poorest americans," August 2021, [Online; posted 27-July-2021]. [Online]. Available: https://www.theatlantic.com/ideas/archive/2021/07/how-race-plays-tax-policing/619570/
[9] C. Thomas and A. Pontón-Núñez, "Automating judicial discretion: How algorithmic risk assessments in pretrial adjudications violate equal protection rights on the basis of race," *Minnesota Journal of Law & Inequality*, vol. 40, no. 2, p. 371, 2022.
[10] R. Angell, B. Johnson, Y. Brun, and A. Meliou, "Themis: Automatically testing software for discrimination," in *Proceedings of the 2018 26th ACM Joint meeting on european software engineering conference and symposium on the foundations of software engineering*, 2018, pp. 871–875.
[11] S. Galhotra, Y. Brun, and A. Meliou, "Fairness testing: testing software for discrimination," in *FSE*, 2017, pp. 498–510.
[12] A. Aggarwal, P. Lohia, S. Nagar, K. Dey, and D. Saha, "Black box fairness testing of machine learning models," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2019, pp. 625–635.

[13] A. Sharma and H. Wehrheim, "Automatic fairness testing of machine learning models," in *IFIP International Conference on Testing Software and Systems*. Springer, 2020, pp. 255–271.

[14] M. Hardt, E. Price, and N. Srebro, "Equality of opportunity in supervised learning," in *NIPS*, 2016.

[15] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel, "Fairness through awareness," in *Proceedings of the 3rd innovations in theoretical computer science conference*, 2012, pp. 214–226.

[16] M. Fan, W. Wei, W. Jin, Z. Yang, and T. Liu, "Explanation-guided fairness testing through genetic algorithm," in *Proceedings of the 44th International Conference on Software Engineering*, ser. ICSE '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 871–882. [Online]. Available: https://doi.org/10.1145/3510003.3510137

[17] P. Zhang, J. Wang, J. Sun, G. Dong, X. Wang, X. Wang, J. S. Dong, and T. Dai, "White-box fairness testing through adversarial sampling," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 2020, pp. 949–960.

[18] H. Zheng, Z. Chen, T. Du, X. Zhang, Y. Cheng, S. Ti, J. Wang, Y. Yu, and J. Chen, "Neuronfair: Interpretable white-box fairness testing through biased neuron identification," in *2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE)*, 2022, pp. 1519–1531.

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

[20] M. Glymour, J. Pearl, and N. P. Jewell, *Causal inference in statistics: A primer*. John Wiley & Sons, 2016.

[21] Q. Guo, X. Xie, Y. Li, X. Zhang, Y. Liu, X. Li, and C. Shen, "Audee: Automated testing for deep learning frameworks," in *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2020, pp. 486–498.

[22] A. Agarwal, P. Lohia, S. Nagar, K. Dey, and D. Saha, "Automated test generation to detect individual discrimination in ai models," *arXiv preprint arXiv:1809.03260*, 2018.

[23] J. Chakraborty, S. Majumder, and T. Menzies, "Bias in machine learning software: Why? how? what to do?" in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2021. New York, NY, USA: Association for Computing Machinery, 2021, p. 429–440. [Online]. Available: https://doi.org/10.1145/3468264.3468537

[24] A. Rényi *et al.*, "On measures of entropy and information," in *Proceedings of the fourth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 547-561. Berkeley, California, USA, 1961.

[25] G. Smith, "On the foundations of quantitative information flow," in *Foundations of Software Science and Computational Structures*, L. de Alfaro, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 288–302.

[26] M. Backes, B. Köpf, and A. Rybalchenko, "Automatic discovery and quantification of information leaks," in *2009 30th IEEE Symposium on Security and Privacy*. IEEE, 2009, pp. 141–153.

[27] Y. Noller and S. Tizpaz-Niari, "Qfuzz: Quantitative fuzzing for side channels," ser. ISSTA 2021. New York, NY, USA: Association for Computing Machinery, 2021, p. 257–269. [Online]. Available: https://doi.org/10.1145/3460319.3464817

[28] A. Aggarwal, P. Lohia, S. Nagar, K. Dey, and D. Saha, "Black box fairness testing of machine learning models," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2019, 2019, p. 625–635. [Online]. Available: https://doi.org/10.1145/3338906.3338937

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

[30] J. Pearl, *Causality*. Cambridge university press, 2009.

[31] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/census+income

[32] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical programming*, vol. 45, no. 1, pp. 503–528, 1989.

[33] H. V. Pham, T. Lutellier, W. Qi, and L. Tan, "Cradle: cross-backend validation to detect and localize bugs in deep learning libraries," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 2019, pp. 1027–1038.

[34] ProPublica, "Compas software ananlysis," https://github.com/propublica/compas-analysis, 2021, online.

[35] "UCI:statlog (german credit data) data set," 2000. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)

[36] "UCI:default of credit card clients data set," 2009. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients

[37] "UCI:heart disease data set," 2001. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Heart+Disease

[38] "Bank marketing uci," 2017. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/bank+marketing

[39] "UCI: Diabetes patient records," 1994. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/diabetes

[40] "Student performance data set," 2014. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Student+Performance

[41] "Medical expenditure panel survey," 2014. [Online]. Available: https://meps.ahrq.gov/mepsweb/

[42] R. K. Bellamy, K. Dey, M. Hind, S. C. Hoffman, S. Houde, K. Kannan, P. Lohia, J. Martino, S. Mehta, A. Mojsilović *et al.*, "AI fairness 360: An extensible toolkit for detecting and mitigating algorithmic bias," *IBM Journal of Research and Development*, vol. 63, no. 4/5, pp. 4–1, 2019.

[43] J. Chakraborty, S. Majumder, Z. Yu, and T. Menzies, "Fairway: a way to build fair ml software," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020, pp. 654–665.

[44] J. M. Zhang and M. Harman, ""ignorance and prejudice" in software fairness," in *43rd IEEE/ACM International Conference on Software Engineering, ICSE 2021, Madrid, Spain, 22-30 May 2021*. IEEE, 2021, pp. 1436–1447. [Online]. Available: https://doi.org/10.1109/ICSE43902.2021.00129

[45] S. Tizpaz-Niari, A. Kumar, G. Tan, and A. Trivedi, "Fairness-aware configuration of machine learning libraries," in *44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022*. IEEE, 2022, pp. 909–920. [Online]. Available: https://doi.org/10.1145/3510003.3510202

[46] Z. Chen, J. M. Zhang, F. Sarro, and M. Harman, "Maat: A novel ensemble approach to addressing fairness and performance bugs for machine learning software," ser. ESEC/FSE 2022, 2022, p. 1122–1134. [Online]. Available: https://doi.org/10.1145/3540250.3549093

[47] V. Nair, Z. Yu, T. Menzies, N. Siegmund, and S. Apel, "Finding faster configurations using flash," *IEEE Transactions on Software Engineering*, vol. 46, no. 7, pp. 794–811, 2020.

[48] Q. Guo, X. Xie, Y. Li, X. Zhang, Y. Liu, X. Li, and C. Shen, "Audee: Automated testing for deep learning frameworks," in *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2020, pp. 486–498.

[49] B. H. Zhang, B. Lemoine, and M. Mitchell, "Mitigating unwanted biases with adversarial learning," in *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, 2018, pp. 335–340.

[50] F. Kamiran, A. Karim, and X. Zhang, "Decision theory for discrimination-aware classification," in *2012 IEEE 12th International Conference on Data Mining*, 2012, pp. 924–929.

[51] A. Agarwal, A. Beygelzimer, M. Dudík, J. Langford, and H. Wallach, "A reductions approach to fair classification," in *International Conference on Machine Learning*. PMLR, 2018, pp. 60–69.

[52] A. Albarghouthi, L. D'Antoni, S. Drews, and A. V. Nori, "Fairsquare: Probabilistic verification of program fairness," no. OOPSLA, 2017. [Online]. Available: https://doi.org/10.1145/3133904

[53] J. Wang, Y. Li, and C. Wang, "Synthesizing fair decision trees via iterative constraint solving," in *Computer Aided Verification: 34th International Conference, CAV 2022, Haifa, Israel, August 7–10, 2022, Proceedings, Part II*. Springer, 2022, pp. 364–385.

[54] T. Piketty and E. Saez, "Income inequality in the united states, 1913–1998," *The Quarterly journal of economics*, vol. 118, no. 1, pp. 1–41, 2003.

[55] E. Black, H. Elzayn, A. Chouldechova, J. Goldin, and D. Ho, "Algorithmic fairness and vertical equity: Income fairness with irs tax audit models," in *2022 ACM Conference on Fairness, Accountability, and Transparency*, ser. FAccT '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1479–1503. [Online]. Available: https://doi.org/10.1145/3531146.3533204

[56] S. Tizpaz-Niari, V. Monjezi, M. Wagner, S. Darian, K. Reed, and A. Trivedi, "Metamorphic testing and debugging of tax preparation software," 2023. [Online]. Available: https://arxiv.org/abs/2205.04998