Uncovering Discrimination Clusters: Quantifying and Explaining Systematic Fairness Violations

Ranit D. Akash
University of Illinois Chicago, USA
rakas@uic.edu

Ashish Kumar
Pennsylvania State University, USA
azk640@psu.edu

Verya Monjezi
University of Illinois Chicago, USA
vmonj@uic.edu

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

Gang Tan
Pennsylvania State University, USA
gtan@psu.edu

Saeid Tizpaz-Niari
University of Illinois Chicago, USA
saeid@uic.edu

***Abstract*—Fairness in algorithmic decision-making is often framed in terms of *individual fairness*, which requires that similar individuals receive similar outcomes. A system violates individual fairness if there exists a pair of inputs differing only in protected attributes (such as race or gender) that lead to significantly different outcomes—for example, one favorable and the other unfavorable. While this notion highlights isolated instances of unfairness, it fails to capture broader patterns of *clustered discrimination* that may affect entire subgroups.**

We introduce and motivate the concept of *discrimination clustering*, a generalization of individual fairness violations. Rather than detecting single counterfactual disparities, we seek to uncover regions of the input space where small perturbations in protected features lead to k-*significantly distinct clusters* of outcomes. That is, for a given input, we identify a local neighborhood—differing only in protected attributes—whose members' outputs separate into many distinct clusters. These clusters reveal significant arbitrariness in treatment solely based on protected attributes, exposing patterns of algorithmic bias that elude pairwise fairness checks.

We present HyFair, a hybrid technique that combines formal symbolic analysis (via SMT and MILP solvers) to certify individual fairness with randomized search to discover discriminatory clusters. This combination enables both formal guarantees—when no counterexamples exist—and the detection of severe violations that are computationally challenging for symbolic methods alone. Given a set of inputs exhibiting high k-discrimination, we further introduce a novel explanation method that generates interpretable, decision-tree-style artifacts.

Our experiments show that HyFair outperforms state-of-the-art fairness verification and local explanation methods. It reveals that some benchmarks exhibit substantial discrimination clustering, while others show limited or no disparities with respect to protected attributes. It also provides intuitive explanations that support understanding and mitigation of unfairness.

I. Introduction

The availability of big data, performant training algorithms, and specialized hardware has made deep feed-forward neural networks (DNNs) [1] a foundational component of modern software systems. DNNs are now routinely deployed in socio-economic decision-making tasks, including risk assessment for criminal reoffense [2], hiring and recruitment [3], income prediction for loans [4], and facial recognition [5]. However, these models are often opaque and highly non-linear, making them prone to unjustified disparities—cases where inputs differing only in protected attributes (e.g., race, gender) yield significantly different outputs. Such disparities raise serious concerns about fairness, particularly in high-stakes domains where automated systems may disproportionately allocate opportunities or resources across social groups.

Traditional fairness verification methods [6], [7] attempt to certify the absence of unfairness through exhaustive search, while fairness testing approaches [8], [9], [10], [11], [12], [13], [14] rely on randomized exploration to uncover counterexamples. However, verification can be computationally prohibitive for rich fairness specifications, and testing often struggles in regions of the input space with low gradient signals or fairness plateaus. Moreover, most existing work focuses on isolated violations of individual fairness and fails to capture broader patterns of systematic bias.

In this work, we propose a new framework for *discrimination clustering*, which generalizes individual fairness violations by uncovering k-significantly distinct clusters of outcomes within counterfactual neighborhoods—regions of the input space differing only in protected attributes. Our hybrid approach, HyFair, combines formal symbolic analysis with randomized search to both certify fairness and quantify the strength of unfairness. By doing so, we aim not just to detect fairness violations but to quantify and explain the underlying systematic disparities that may otherwise remain hidden.

Modeling Fairness Violations. When analyzing decision-support software through the lens of fairness, it is common to model such systems as binary classifiers, where outputs are categorized as either *favorable* or *unfavorable*. The features of inputs under consideration can be partitioned into *protected attributes* (e.g., race, gender identity, disability status) and *non-protected attributes* (e.g., income, work experience, education level). Under a standard formulation of equality of opportunity [15], fairness requires that decision outcomes depend only on relevant, non-protected features so that similar individuals, differing only in protected attributes, receive similar outcomes. This notion underlies the search for individual discrimination (ID), where an individual and their counterfactual—differing only in a protected attribute—receive different outcomes. Such

2643-1572/25/$31.00 ©2025 IEEE
DOI 10.1109/ASE63991.2025.00141

individual fairness violations, also known as discriminatory instances, have been widely studied [8], [9], [10], [11], [12], particularly in the software testing community. However, this binary framing overlooks complex patterns of unfairness, such as when neighborhoods of inputs, differing only in protected attributes, lead to multiple divergent and arbitrary outcomes.

To capture these richer fairness violations, we propose a quantitative generalization of counterfactual fairness. We define a system to be k-discriminant if, within a group of K counterfactual inputs—records that differ only in protected attributes—the system produces $2 \leq k \leq K$ distinct outcomes. Much like how k-means clustering partitions data based on feature similarity, this formulation identifies clusters of discriminatory behavior: localized regions in the input space where variations in protected features alone lead to outcome groupings that are meaningfully separated. We refer to this phenomenon as *discrimination clustering*. This notion offers multiple advantages over prevalent individual discrimination. While exiting tools can generate hundreds of thousands of IDs, k-discrimination allows us to prioritize test cases based on the severity of discrimination. Finding k-discriminants also helps uncover worst-case scenarios where a DNN makes highly inconsistent (or arbitrary) decisions for similar individuals based on their protected attributes. These instances often show a structured pattern of systematic disparities [16] that moves beyond isolated violations of individual fairness.

HYFAIR: Characterizing k-Discriminant Clusters. Given the pervasiveness of their socio-economic-critical applicability, fairness testing and verification for DNNs have received considerable attention [17], [18], [19], [20]. Verification approaches [6], [7], [21] aim to provide a mathematical proof of fairness through exhaustive exploration, while testing-based approaches [8], [9], [10], [11], [12], [13], [14] seek to increase trust by randomized discovery of discrimination-related bugs. Exhaustive formal search excels at proving the absence of violations but struggles to scale for complex discrimination properties, and may generate large volumes of counterexamples, overwhelming human analysts. In contrast, randomized search scales better but often performs poorly when exploring flat or low-signal regions of the outcome space.

Both strategies offer complementary strengths. Randomized approaches scale well and have been effective in uncovering 2-discriminant instances (individual fairness violations), but they cannot guarantee the absence of discrimination. As Dijkstra might put it, testing can show the presence of discrimination, but never its absence. Meanwhile, the formal methods community [6], [7], [22], [21] has advanced verification approaches using constraint solvers (e.g., abstract interpretation, SMT) to exhaustively prove the absence of individual discrimination, i.e., the lack of 2-discriminant counterfactuals. However, such approaches face scalability challenges when generalized to quantitative fairness, such as k-discriminant clusters.

This paper presents HYFAIR*, a hybrid approach for the fairness analysis of DNNs that combines formal verification with randomized exploration to uncover, quantify, explain, and mitigate* clustered patterns of discrimination*. The goal is to support developers of data-driven software in diagnosing and addressing fairness violations that arise from significant arbitrariness in the behavior of DNNs.*

Our hybrid approach first uses MILP solvers [6], [23] to either certify fairness or find counterexamples witnessing 2-discriminant instances. Our results show that HYFAIR significantly outperforms FAIRFY [6], the state-of-the-art technique in terms of finding more individual discrimination instances quicker (RQ1). It then performs a local randomized search around these seed counterexamples to identify maximum k-discriminant clusters, i.e., structured regions in the input space where protected attribute perturbations lead to multiple distinct outcomes. We use a simulated annealing search to find inputs with the maximum k-discrimination that outperforms baseline randomized search strategies (RQ2).

HYFAIR: Debugging k-Discriminant Clusters. We next aim to understand the common conditions underlying the set of edge-case inputs $\{\mathbf{a}_1, \ldots, \mathbf{a}_k\}$ that witness a k-discriminant cluster, i.e., a localized region where protected attribute perturbations yield significantly divergent outcomes. While these k inputs jointly expose a failure of fairness, existing explanation techniques fall short: differential debugging [24], [25] compares faulty and passing traces pairwise, and eXplainable AI (XAI) methods [26], [27], [28], [29] typically focus either on single-instance local explanations or on global approximations of the model. To address this gap, we introduce a novel explanation framework tailored to discrimination clustering. Our method leverages local perturbations and decision tree learning to uncover logical conditions that are shared across members of the k-discriminant set. Our experiments show its efficacy against the baseline XAI methods (RQ3). These explanations also help mitigate unfairness in the DNNs (RQ4).

Contributions. This paper makes the following contributions.

1) A novel fairness notion, k-discriminant, is introduced to characterize bias through structured clusters of discriminatory outcomes in deep neural networks.
2) A *mixed-integer linear programming* (MILP) based approach is developed to certify individual fairness and guide targeted randomized search.
3) A novel *explainable AI method* is proposed that leverages decision-tree learning and local perturbations to identify the root causes of k-discriminant clusters.
4) HYFAIR, a hybrid framework, is presented that integrates formal verification, randomized search, and explanation techniques to detect, explain, and mitigate discrimination clustering in DNNs.

II. OVERVIEW

We focus on the notion of *individual fairness* [17], where fairness is violated if altering protected attributes, while keeping all other features fixed, changes the machine learning outcome from favorable to unfavorable (or vice versa). Inspired by `DICE` [20], we define a model as k-discriminant (or k-unfair)

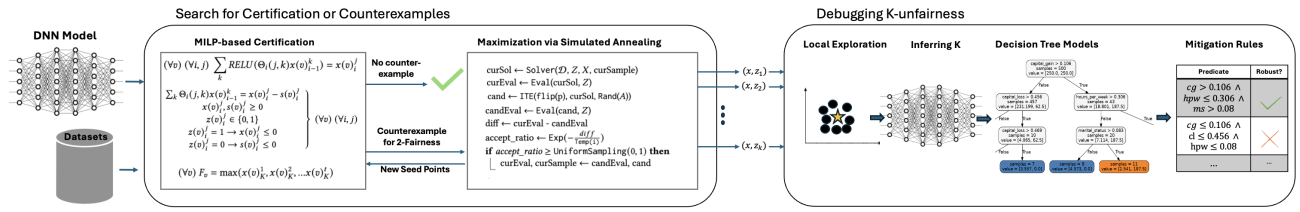


Fig. 1: HYFAIR Framework.

if it produces k distinct outcomes for a set of K inputs ($k \leq K$) that differ only in their protected attributes.

Real-world implications and advantages of k-discriminant. Our proposed generalization from the individual discrimination notion [30] to k-discriminant acknowledges that fairness is rarely binary and provides a rigorous mathematical foundation for evaluating discrimination in complex socio-technological systems. One key implication is discovering significant "arbitrariness" in decision-making. This notion can reveal when k similar individuals receive significantly different likelihood scores, and hence it shows areas where the DNN's decision-making becomes highly arbitrary based on protected attributes. Consider a loan application scenario where an unprivileged applicant is denied. When k=10, we found cases where ten applicants with nearly identical qualifications (e.g., credit scores within 1% range, same income bracket, similar debt ratios) received vastly different loan approval probabilities based on their backgrounds. Individual fairness would only compare pairs and miss this systematic arbitrariness. By identifying areas of maximum arbitrariness, our explanation technique can pinpoint the specific combinations of features that trigger irrational decision-making.

HYFAIR Workflow Summary. Figure 1 summarizes the workflow of HYFAIR. Given a dataset and a pre-trained DNN model, HYFAIR operates in two phases. In the *search* phase, it encodes the DNN as a mixed-integer linear program (MILP) and verifies it against individual fairness. If a counterexample is found (a 2-discriminant), HYFAIR initiates a randomized search—using both random walks and simulated annealing—to characterize the maximum k-discrimination starting from the counterexample. HYFAIR alternates between verification and search, using the MILP solver to generate new random seeds for the search procedure, helping it escape potential plateaus in the input space. For example, on the *Adult Census Income* dataset [31] (AC2 benchmark with $K = 90$), HYFAIR identified 18.6 ($\pm$ 6.7) 2-discriminants on average, while FAIRFY [6] found only 0.6 ($\pm$ 0.5). The average maximum k was 17.8 ($\pm$ 0.4) for HYFAIR compared to 8.7 ($\pm$ 5.4) for FAIRFY. Notably, only 3.8 ($\pm$ 2.3) of the 2,245 counterexamples exhibited the maximum k-discrimination of 19, highlighting the rarity and significance of these bugs.

In the *debugging* step, given a set of inputs that exhibit maximum k-discrimination, we first explore their local neighborhood via random sampling and query the DNN to assess how k-discriminant behavior generalizes across nearby inputs. We then train a decision tree to explain the conditions under which the DNN exhibits clustered discrimination. The resulting model yields a set of logical predicates that evaluate to `true` when the DNN significantly discriminates.

To understand the causal influence of these predicates, we evaluate whether flipping the conditions in the explanation models can reduce the observed k—effectively identifying input subspaces where mitigating the predicate leads to fairer behavior. We then use these rules to implement guardrails that constrain the DNN's behavior, and we leverage the associated test cases to fine-tune the model for mitigation. Compared to LIME [26], HYFAIR provides more robust and succinct explanations with broader input coverage. On the AC-2 benchmark, our guided mitigation reduced the number of discriminatory instances from 2,245.0 to 769.6 and lowered the success rate of fairness bug reproduction from 87.7% to 36.3%.

III. THE DISCRIMINATION CLUSTERING PROBLEM

In this study, we analyze deep neural network classifiers that have undergone prior training (i.e., pre-trained DNN).

Definition III.1 (DNN: Interpretation)**.** *A deep neural network (DNN) defines a function $F : X \times Z \to [0,1]^t$, where $X = X_1 \times X_2 \times \ldots \times X_n$ denotes the space of non-protected input attributes (e.g., occupation, income, education), and $Z = Z_1 \times Z_2 \times \ldots \times Z_m$ denotes the space of protected input attributes (e.g., race, gender, age). The output of F is a t-dimensional probability vector over class labels. For any input (x, z), the predicted class label is given by*

$$F_{label}(x, z) = \arg\max_{i \in [t]} F(x, z)(i).$$

We assume the domain of the protected attribute space Z is finite, with K denoting the number of distinct protected groups.

Definition III.2 (DNN: Structure)**.** *A DNN F is defined by its input dimension $n + m$, output dimension t, number of hidden layers N, and weight matrices $\Theta_1, \Theta_2, \ldots, \Theta_N$. We analyze a pre-trained DNN with fixed parameters and weights to assess its fairness. For each layer $r \in \{1, \ldots, N\}$, the output $\mathcal{D}_r$ is computed as an affine transformation of the previous layer's output $\mathcal{D}_{r-1}$ using weights Θ_r, followed by a non-linear activation. Specifically:*

1) *For hidden layers $1 \leq r < N$, the activation is a ReLU function, defined as $\mathcal{D}_r = \max\{\Theta_r \cdot \mathcal{D}_{r-1}, 0\}$.*
2) *For the output layer $r = N$, a SoftMax function maps the final linear outputs to a probability distribution over the t classes.*

The term $\mathcal{D}_r^j$ denotes the output of neuron j in layer r.

The 2-Discriminant Problem. The prevailing notion of fairness, e.g., individual discrimination [32], [10], [11], requires that similar individuals—who differ only in protected attributes—receive similar outcomes. Formally, a DNN is said to be 2-*discriminant* if there exist two protected attributes $z_1 \neq z_2 \in Z$ and a shared unprotected input $x \in X$ such that:

$$\texttt{Dist}_\epsilon\big(F(x, z_1), F(x, z_2)\big) > \epsilon,$$

where $\texttt{Dist}_\epsilon$ measures the deviation between the outputs of the DNN for inputs differing only in protected attributes, and ϵ is a specified tolerance threshold. Conversely, we say the DNN satisfies *individual fairness* if no such counterexample exists; that is: $\forall x \in X, \forall z_1, z_2 \in Z, \texttt{Dist}_\epsilon\big(F(x, z_1), F(x, z_2)\big) \leq \epsilon$. In this case, the DNN is said to be 2-*fair*, as no pair of counterfactuals differing only in protected attributes yields an output difference greater than ϵ.

Search for k-Discrimination. From an AI risk perspective, we are often required to reason beyond 2-discriminant behavior and assess the *maximum unfairness* exhibited by a DNN model. We say a model is k-*discriminant* if there exists a set of K inputs $(x, z_1), (x, z_2), \ldots, (x, z_K)$, where all records share the same non-protected features $x \in X$ but differ in protected attributes $z_i \in Z$, and the model produces $k \leq K$ *distinct* outputs. Formally:

$$\exists z_1, \ldots, z_K \in Z,\ x \in X \text{ s.t. } \mathtt{C}_\epsilon\big(F(x, z_1), \ldots, F(x, z_K)\big) == k,$$

where $\mathtt{C}_\epsilon(\cdot)$ is a clustering function that partitions the outputs into $1 \leq k \leq K$ groups based on an indistinguishability threshold ϵ. Since the output of F lies in the range $[0, 1]$, the threshold ϵ can be used to define uniform partitions of the outcome space for clustering. We note that if the model is 2-fair (i.e., not 2-discriminant), then it is trivially k-fair for all $k \geq 2$. Otherwise, once a 2-discriminant counterexample is found, we aim to compute the *maximum value of k-discrimination* by solving:

$$\max_{x \in X,\ z_1, \ldots, z_K \in Z} \mathtt{C}_\epsilon\big(F(x, z_1), \ldots, F(x, z_K)\big).$$

To solve the k-discriminant problem, we construct up to $k+1$ copies of the DNN, denoted $\mathcal{F} = F_1 \times \ldots \times F_{k+1}$, where each copy is evaluated on the same non-protected attributes but different protected attributes. Verifying fairness at level k involves checking whether the outputs of these $k+1$ copies can be clustered into fewer than $k+1$ distinct groups. Thus, increasing values of k require more copies of the model, making the verification problem increasingly expensive.

While formally verifying the absence of k-discrimination becomes computationally intractable as k grows, scalable techniques exist for certifying the 2-discriminant property [6], [23]. Our key insight is to leverage these solvers not just for certification, but also as a foundation to guide the quantification of the model's maximum k-discrimination.

Debugging for k-Discrimination. We now turn to understanding the common conditions among edge-case inputs $\mathcal{D} = \{(x, z_1), \ldots, (x, z_K)\}$ that exhibit significant discriminatory behavior in a DNN. Since this set collectively witnesses k-discrimination, standard methods such as differential debugging [24], [25] (comparing faulty vs. passing traces), local explanations [26] (explaining a single instance), and global explanation techniques [27] (summarizing model behavior over the entire input space) fall short in explaining the root causes of such clustered discrimination.

Our goal is to identify conditions on non-sensitive attributes under which the model becomes disproportionately sensitive to protected attributes in its decision-making. The explanation challenge is to uncover what properties distinguish significantly discriminatory instances from benign ones.

Discrimination Clustering Problem. Given a pre-trained DNN model F with protected attributes Z ($|Z| = K$) and non-protected attributes X, and a formal computational model of F that can certify 2-fairness using a distance relation $\texttt{Dist}_\epsilon$, our goal is to:

(i) certify the DNN model F against 2-fairness property,
(ii) determine the maximum $k \in [2, K]$ for which the model exhibits significant k-discrimination, and
(iii) explain/mitigate the root causes of k-discrimination.

IV. HyFair for Discrimination Analysis

We note that if no counterexample exists to the 2-fairness property across all protected attributes, then the DNN is k-fair for all $k \geq 2$. However, if a 2-discriminant counterexample is found, the model violates individual fairness, and the goal is to characterize the extent of this violation by identifying the maximum value of k or which the model is k-discriminant.

We address this in two phases. First, we formulate the 2-fairness verification problem using symbolic reasoning techniques such as mixed-integer linear programming (MILP) and satisfiability modulo theories (SMT). These solvers can either certify the model's fairness or generate counterexamples that serve as individual discriminatory instances. Second, to assess the severity and structure of discrimination, we employ randomized search strategies that explore neighborhoods around these counterexamples to identify the maximum k-discrimination witnessed by the model.

Certifying 2-Fairness requirements. Unlike adversarial robustness, which is typically a local property around a specific input, individual fairness is a global property: any two inputs differing only in protected attributes—regardless of their location in the input space—must yield similar outputs. Therefore, verification approaches for local robustness [33], [34] are insufficient for certifying individual fairness.

SMT Solver. FAIRIFY [6] formulates the individual fairness problem with an SMT-based approach to verify individual fairness property in DNN. We briefly summarize this approach below. Let a DNN be viewed as a function $F : \mathbb{R}^m \to \mathbb{R}^n$ and let F_1 and F_2 be two copies of the same DNN, then any fairness property can be formulated as the verification query:

$$\phi_{pre}(x, x') \wedge \phi_{dom}(x, x') \wedge y = F_1(x) \wedge y' = F_2(x') \rightarrow \phi_{post}(y, y')$$

where x and x' are the inputs to F_1 and F_2 respectively, y and y' are the outputs to F_1 and F_2 respectively, ϕ_{pre} is the precondition clause on the two inputs for the given fairness property (e.g., x and x' don't differ on the non-protected attributes), ϕ_{dom} is the domain constraints on the input and ϕ_{post} is the postcondition clause on the two outputs for the given fairness property (e.g., y and y' are equal). Such a verification query can be fed into an SMT solver to check for satisfiability, and furthermore, can also be asked to construct counterexamples if it is unsatisfiable.

Mixed Integer Linear Programs. We use a MILP encoding, similar to [23], [35], [36] to certify individual fairness of DNN. We describe the MILP formulation of our DNN inspired by [23]. We have two copies of a pertained DNN each with R hidden layers, input dimension $n+m$, and output dimension t. Let the first n dimensions of the input be the non-protected input and the latter m dimensions be the protected input. If $x(v)_r^j$ denotes the output of the jth neuron in the rth layer for the v^{th} DNN and Θ_i denotes the weight for layer i for both the DNNs, then the computation equations for our DNNs are:

$$(\forall v)\ (\forall i, j)\ \sum_r \texttt{ReLU}(\Theta_i(j,r)x(v)_{i-1}^r) = x(v)_i^j$$

We can denote these constraints in linear form by introducing new variables $s(v)_r^j$ and $z(v)_r^j$, which represent the negative output of the jth neuron in the rth layer and the activation state of the ReLU unit acting on the jth neuron in the rth layer respectively. Noting that $\texttt{ReLU}(x) := \max(x, 0)$, we get our corresponding linear constraints to be: for all v, i and j:

$$\begin{aligned}
\sum_r \Theta_i(j,r)x(v)_{i-1}^r &= x(v)_i^j - s(v)_i^j \\
x(v)_i^j, s(v)_i^j &\geq 0 \\
z(v)_i^j &\in \{0,1\} \\
z(v)_i^j = 1 &\rightarrow x(v)_i^j \leq 0 \\
z(v)_i^j = 0 &\rightarrow s(v)_i^j \leq 0
\end{aligned}$$

Additionally, our input to both DNNs is identical on the non-protected inputs, i.e.,

$$(\forall j \leq n)\ x(1)_0^j = x(2)_0^j$$

for $j \in \{1, \ldots, n\}$. If we denote the final output of the DNNs by F_1 and F_2 respectively, then F_v is given by the maximum of the output of all neurons in layer R i.e.

$$(\forall v)\ F_v = \max(x(v)_R^1, x(v)_R^2, ...x(v)_R^t)$$

As before we can denote these constraints in linear form by introducing auxiliary variables. Finally, our optimization objective is given by $\max |F_1 - F_2|$. Our objective function can be made linear by noting that $|x| = \max(x, -x)$ and then using the techniques shown before to convert the *max* function into a linear form by introducing auxiliaries. Finally, it is trivial to see that our two DNNs satisfy the 2-Fairness property iff the output of the corresponding MILP is at most ϵ (we use ϵ of 0.05 similar to FAIRIFY [6]).

Algorithm 1: HYFAIR SEARCH

Input: Deep learning model $\mathcal{D}$, Test Data Samples A, protected attributes Z, Non-Protected attributes X, Formal computation model `Solver`, An evaluation function `Eval`, Type of search τ: 'RW', 'SA', or 'SA+KNN', Temperature of SA temp, head probability p, and Time-out T.

Output: bestEval, bestidx

```
seed, i, bestEval ← Rand(A), 0, 0
curSol ← Solver(D, Z, X, seed)
curEval ← Eval(curSol, Z)
while Not Time-out(T) do
    if curEval > bestEval then
        bestEval, bestSol ← curEval, curSol
    else
        curSol, curEval ← Solver(D, Z, X, curSol),
          Eval(curSol, Z)
    if τ == RW then
        cand ← RandNeighbor(A, curSol)
        curSol, curEval ← cand, Eval(cand, Z)
    else
        if τ == SA then
            cand ← ITE(flip(p), curSol, Rand(A))
        else if τ == SA+KNN then
            cand ← ITE(flip(p), KNN(curSol),
              Rand(A))
        candEval ← Eval(cand, Z)
        diff ← curEval - candEval
        accept_ratio ← Exp(-diff/Temp(i))
        if accept_ratio ≥ UniformSampling(0,1)
          then
            curSol, curEval ← cand, candEval
    i ← i + 1
return (bestEval, bestIdx)
```

Finding k-Discriminants with Randomized Search. Algorithm 1 takes as input a deep learning model $\mathcal{D}$; a dataset A consisting of protected attributes P and non-protected attributes Q; a `Solver` (either `SMT` or `MILP`); an `Eval` function that measures the fitness of the solution against the K-fairness criterion; a search type τ (e.g., random walk, simulated annealing (SA), or a hybrid SA with nearest-neighbor heuristics); a temperature function for SA; and a timeout T. The algorithm outputs a solution that witnesses a violation of the k-fairness requirement. Following FAIRIFY [6], we set ϵ=0.05, meaning the DNN's outputs are considered distinct if they differ by more than 5% in predicted score as the protected attributes vary while others remain fixed.

Initialization: A random data sample from the dataset is selected as the current seed to query `MILP` or `Z3` solvers. We then use the current seed to search the DNN $\mathcal{D}$ via the

Algorithm 2: HYFAIR DEBUGGING

Input: DNN model $\mathcal{D}$; k-discriminant input set $X = \{(x, z_1), \ldots, (x, z_K)\}$; number of neighborhood samples n; high k-discrimination threshold κ; significant difference threshold δ.

Output: Explanation predicates ϕ

1 $X_{\text{perturb}} \leftarrow \texttt{localPerturbation}(X, n)$ `// Generate neighborhood samples`

2 $\text{Scores} \leftarrow \mathcal{D}(X_{\text{perturb}})$ `// Evaluate perturbed inputs`

3 $Y_{\text{perturb}} \leftarrow \texttt{getHighLowKLabels}(\text{Scores}, \kappa)$ `// Label based on k`

4 $\Phi \leftarrow \texttt{buildDecisionTree}(X_{\text{perturb}}, Y_{\text{perturb}})$ `// Train decision tree`

5 $L, \Pi \leftarrow \texttt{getLeavesAndPaths}(\Phi)$ `// Extract leaves and paths`

6 **foreach** $(l, \pi) \in (L, \Pi)$ **do**

7 **if** $\texttt{isHighKLeaf}(l)$ **then**

8 $X_{\text{cex}} \leftarrow \texttt{sampleCounterExamples}(\pi)$

9 $\text{Scores}_{\text{cex}} \leftarrow \mathcal{D}(X_{\text{cex}})$

10 $\text{meanKDiff} \leftarrow \texttt{getKStats}(\text{Scores}, \text{Scores}_{\text{cex}})$

11 **if** *meanKDiff* $\geq \delta$ **then**

12 $\phi.\texttt{add}(\pi)$

13 $\mathcal{D}' \leftarrow \texttt{train}(\mathcal{D}, X, \phi)$ `// Retrain with explanation-based mitigation`

14 **return** $\phi, \mathcal{D}'$

`Solver` that finds a 2-discriminant. We take the instance and query the DNN over all possible protected values to measure the number of buckets for the evaluation of instances.

Iterations: The loop executes until the timeout T is reached. In each iteration, we first check if the current solution gives the best solution. Otherwise, we might be stuck in the local minima, hence we query the `Solver` to get a new solution. Depending on the type of search (discussed below), we generate a candidate sample from the current solution (derived by the Solver) and accept it as the current sample.

Search Type: The logic of search significantly depends on the type of search τ. We consider the following search strategy:

1) **Stateless Random Search**. We randomly choose a data point from the neighborhood of the current solution and accept it as the next sample to explore. Since this approach navigates the search space uniformly at random without any guidance; this serves as the baseline.
2) **Simulated Annealing Search**. We consider the current solution, inferred by the solver, with the probability p and a random data point from A with the probability $1-p$ as the candidate for some large $p \geq 0.9$. Then, we follow the Metropolis algorithm, where we evaluate the fitness of the candidate and accept it as the next sample with the probability that is proportional to its difference from the fitness of the current solution.
3) **Simulated Annealing with Nearest Neighbor Search**. This type of search uses nearest neighbors of current solutions in addition to a random sample from A. Specifically, the algorithm uniformly chooses one of the nearest neighbors with the probability p and a random data point from A with the probability $1-p$ as the candidate for some large $p \geq 0.9$.

Debugging k-discrimination. Given the set of k samples that witness the maximum k-discrimination in the DNN, i.e., $\mathcal{D} = \{(x, z_i)\}_{i=1}^{K}$, our next goal is to understand the circumstances under which the DNN model becomes arbitrary and come up with a mitigation strategy. One naive idea is to infer K local explanations via methods like LIME [26] and mine common patterns among the K explanatory models. However, this approach might be both expensive and fail to find useful patterns, and derive limited explanations. We need a robust framework to learn a common explanation for all K points together. Our approach is to leverage the decision tree algorithms to synthesize a set of predicate functions $\phi_j : X \rightarrow \mathbb{B}$ such that $\phi_j(x) = \texttt{true}$ provides an explanation about the circumstances over the non-protected attributes that led to the maximum arbitrariness of DNNs.

Algorithm 2 shows our approach to inferring the predicate functions to explain bugs. Given a set of inputs that witnesses significant discrimination, we first generate n neighborhood data samples around the input by local perturbations (Line 1). Then, we query the DNN model to measure k-discrimination for each of those neighborhood data samples (Line 2). To provide a succinct explanation, we convert k values into high and low labels (binary classes of high-K vs. low-K) using the 0.95-percentile of k values, where only the top 5% of k values belong to the high-K class of significantly discriminatory (Line 3). Then, we infer a decision tree model that yields a set of paths (a predicate function) in the hyper-rectangular input space to explain what properties are common inside the high-class inputs and what properties distinguish high and low classes (Line 4). Then, we retrieve each path in the tree and its corresponding leaf node and go through each path and label to identify explanatory models (Line 5-6). If the leave node is a high-K class (Line 7), then the corresponding path in the tree that traverses from the root node to this leaf node is a candidate to explain the significant discriminatory inputs. To add the candidate path to the final set of explanatory models, we sample data points that evaluate the candidate path to `false`, i.e., negating the predicate function (Line 8), and compute the corresponding k values of these instances by querying the DNN model (Line 9). We calculate the difference between the mean of k for the data samples that satisfy the path conditions and those that negate the conditions (Line 10). If the differences are more than a threshold δ, we deem the path robustly explains the significant discriminatory instances (Line 11-12). We use the generated samples and DT predicates to mitigate unfairness [37] in the DNNs (Line 13).

V. EXPERIMENTS

In this paper, we pose the following research questions:

RQ1 How does HYFAIR compared to FAIRFY [6] in certifying 2-fairness and finding counterexamples?

RQ2 What is the performance of different randomized search algorithms in characterizing k-discrimination?

RQ3 What are the performance and complexity of HYFAIR's explanations as compared to the baseline LIME [26]?

RQ4 How does HYFAIR help improve fairness?

Benchmarks. We have 20 DNN benchmarks of various architectures (fully connected and based on ReLU activation functions) from the literature [6]. They include real-world DNNs in Kaggle [38] and the SE literature [39], [9], [32], [40], [41] where the number of layers and neurons vary from 3-11 and 10-318 neurons. These benchmarks are trained over two popular and socially critical datasets. The Adult Census dataset concerns whether an individual earns more than 50K or not. The Bank Marketing dataset is used for the prediction of whether a client will subscribe to a service or not.

Technical Details. We implemented HYFAIR in Python v3.8.10 with TensorFlow v2.12.0 (Keras API) and scikit-learn v1.2.2. 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 take the average of 10 multiple runs for all experiments and report the standard deviations from the mean. We also set K to 20 throughout the experiments.

Hyperparameter Selection. Following FAIRIFY [6], we set the MILP solver timeout to 100 seconds and used the default convergence tolerance of 0.0001. We run the randomized search for 4 hours. While we did not fix a threshold for k-discrimination during the search, we used a 95-percentile threshold to classify samples into high and low k values for training decision trees. We generated 5,000 samples following LIME [26] established methodology, from the neighborhood of high k-value instances to explain circumstances under which the DNNs exhibited arbitrary behaviors. This number balanced the explanation quality with computational efficiency, required to quantify the impact of local samples on model outcomes.

RQ1: Finding individual discrimination. We compare our tool to the FAIRIFY [6], a formal verification method to certify 2-fairness properties of DNNs or find counterexamples as the individual discriminatory instances (2-discriminants). The key difference between HYFAIR and FAIRIFY is that Fairify models the DNNs with SMT formulations and uses Z3-solver whereas HYFAIR models DNNs with MILP formulations.

Table I compares Fairify to our tool HYFAIR. The statistically significant results are highlighted with bold fonts in the tables. Here, we report $\#ID$ - the number of individual discriminatory (ID) instances, $T.1st\ (s)$ - time to the first ID, $\#K.1st$ - k-discrimination for the first ID instance, and $\#K$ - the maximum k-discrimination.

We consider #ID and #K as the metrics for efficacy, and the T.1st as the metric for efficiency. HYFAIR outperforms Fairify in 85% of the cases (17 out of 20 cases) in finding more IDs in the given timeout. In 5 cases, Fairify does not find any ID instances where HYFAIR finds a good number of IDs. Considering #K, HYFAIR outperforms Fairify in 80% of the cases. In terms of efficiency of finding the first solution (i.e., T.1st), HYFAIR outperforms Fairify in 90% of cases. Farify spends a significant amount of time on preprocessing (input partition) and pruning.

We note that in the BM4 and BM8 cases, only one of the tools can find discriminatory instances. This leads us to understand crucial differences between MILP- vs. Z3-based techniques. We find that Z3 produces counterexamples that are often close to each other and almost belong to the same region. On the other hand, MILP produces counterexamples from different regions of the search space, oftentimes from the boundaries that separate a safe region from an unsafe one. Each technique might be effective for different benchmarks.

Answer RQ1: Our MILP formulation significantly outperforms the baseline Fairify in most of the cases (more than 80% of the time) in terms of finding more individual discriminatory (ID) instances in shorter amounts of time.

RQ2: Search for k-discrimination. Our goal is to find instances that witness the maximum k-discrimination. We compare the efficacy and efficiency of two variations of simulated annealing (SA) algorithms in characterizing the amounts of dependencies between protected attributes and DNN outcomes. We also include a Random Walk (RW) to establish a baseline that abolishes the SA.

Table II summarizes the outcomes of this experiment. Since we certify 2-fairness for BM4 and BM7, we exclude them from the search. The columns are similar to Table I, except that $Iter$ shows the number of completed iterations, $succ.rate$ shows the proportion of ID instances relative to the total generated instances during the search, and $\#ID.maxK$ reports the number of IDs with max k-discrimination, and $T.maxK$ shows the time taken to find the max k-discrimination.

To compare the performance of the search algorithms, we focus on the following metrics: $Max.K$, $Avg.K$, $\#ID.maxK$, and $\#T.maxK$. While all three randomized search algorithms effectively identify ID instances, the SA algorithm consistently outperforms the others, identifying ID instances with the maximum buckets in 94% of the cases. When considering $\#ID.maxK$ and $T.maxK$, SA algorithms perform slightly better in 50% of the cases, whereas RW and SA+KNN show better results in 39% and 10% of the cases, respectively. Also, taking into consideration the number of ID instances, the SA algorithm leads, finding the highest number of IDs in 56% of the benchmarks, slightly surpassing the RW algorithm, which achieves this in 50% of cases. Additionally, the SA algorithm demonstrates a clear advantage in terms of time to the first solution, discovering the first ID instance faster than the other algorithms in 72% of the benchmarks. Focusing on the average number of clusters ($Avg.K$), the SA+KNN algorithm outperforms other algorithms by generating ID instances that exhibit a higher average number of clusters in 56% of the benchmarks, where SA and SW algorithms achieve 33% and 17%, respectively.

TABLE I: Comparison of SMT Solver to MILP solver for characterizing k-fairness.

DNN	FAIRFY [6] (SMT)				HYFAIR (MILP)			
	$\#ID$	$T.1st$ (s)	$\#K.1st$	Num. Clusters (#K)	$\#ID$	$T.1st$ (s)	$\#K.1st$	Num. Clusters (#K)
AC1	1.4 (± 1.1)	81.5 (± 70.2)	8.2 (± 4.9)	11.2 (± 5.1)	**23.2** (± 2.9)	**0.5** (± 0.2)	**17.6** (± 0.9)	**18.6** (± 0.6)
AC2	0.6 (± 0.5)	75.2 (± 108.5)	8.7 (± 5.4)	8.7 (± 5.4)	**18.6** (± 6.7)	**2.7** (± 2.2)	**15.8** (± 1.1)	**17.8** (± 0.4)
AC3	2.0 (± 2.8)	32.0 (± 29.9)	9.2 (± 5.7)	11.2 (± 6.6)	**12.6** (± 4.6)	**3.4** (± 2.0)	**11.4** (± 3.0)	**16.0** (± 2.4)
AC4	**0.4** (± 0.9)	**71.2** (± 0.0)	**9.0** (± 6.4)	**12.0** (± 8.5)	**0.4** (± 0.5)	106.2 (± 50.3)	8.5 (± 2.1)	8.5 (± 2.1)
AC5	0.0 (± 0.0)	N/A(± N/A)	N/A(± N/A)	N/A(± N/A)	**4.0** (± 0.7)	**25.1** (± 15.8)	**10.2** (± 4.4)	**15.0** (± 0.0)
AC6	0.4 (± 0.5)	139.1 (± 83.5)	9.5 (± 0.7)	9.5 (± 0.7)	**25.0** (± 1.6)	**0.3** (± 0.0)	**11.8** (± 4.4)	**16.6** (± 3.3)
AC7	0.0 (± 0.0)	N/A(± N/A)	N/A(± N/A)	N/A(± N/A)	**2.0** (± 0.0)	**73.8** (± 31.2)	**15.2** (± 3.3)	**17.2** (± 2.6)
AC8	6.8 (± 2.3)	12.7 (± 7.6)	10.6 (± 2.5)	13.6 (± 2.4)	**25.0** (± 0.7)	**0.2** (± 0.0)	**15.8** (± 0.8)	**17.2** (± 0.4)
AC9	12.8 (± 2.3)	7.4 (± 5.1)	5.6 (± 2.0)	11.6 (± 0.5)	**23.6** (± 1.3)	**0.2** (± 0.0)	**14.8** (± 0.8)	**16.0** (± 0.0)
AC10	0.8 (± 0.8)	101.9 (± 53.6)	6.7 (± 5.7)	9.7 (± 6.7)	**23.4** (± 1.1)	**0.43** (± 0.0)	**15.0** (± 1.6)	**17.6** (± 0.5)
AC11	0.0 (± 0.0)	N/A(± N/A)	N/A(± N/A)	N/A(± N/A)	**0.8** (± **1.3**)	**64.2** (± 53.9)	**8.5** (± 0.7)	**9.0** (± 1.4)
AC12	0.0 (± 0.0)	N/A(± N/A)	N/A(± N/A)	N/A(± N/A)	**13.2** (± **4.4**)	**3.8** (± 2.9)	**15.2** (± 3.2)	**18.8** (± 1.6)
BM1	1.2 (± 1.6)	4.19 (± 0.1)	3.0 (± 0.0)	6.0 (± 4.24)	**37.8** (± 4.4)	**2.4** (± 1.4)	**7.8** (± 0.8)	**9.0** (± 0.0)
BM2	5.0 (± 6.4)	42.6 (± 41.3)	4.2 (± 3.0)	4.4 (± 3.3)	**33.2** (± 11.5)	**5.3** (± 4.8)	**7.4** (± 1.1)	**9.0** (± 0.0)
BM3	4.4 (± 4.0)	22.3 (± 46.7)	4.0 (± 2.3)	5.2 (± 2.5)	**58.8** (± 2.8)	**2.3** (± 1.1)	**7.6** (± 1.1)	**9.0** (± 0.0)
BM4	**0.6** (± **1.3**)	**6.5** (± 0.0)	**2.0** (± 0.0)	**2.0** (± 0.0)	0.0 (± 0.0)	N/A(± N/A)	N/A(± N/A)	N/A(± N/A)
BM5	13.8 (± 12.6)	5.8 (± 10.7)	2.4 (± 0.5)	3.4 (± 0.9)	**43.2** (± 4.8)	**1.6** (± 1.2)	**7.8** (± 1.3)	**9.0** (± 0.0)
BM6	17.2 (± 14.1)	17.7 (± 37.5)	**5.2** (± 2.5)	**8.2** (± 1.1)	**59.8** (± 6.7)	**3.5** (± 2.1)	4.0 (± 1.0)	5.8 (± 0.4)
BM7	**8.4** (± 9.2)	**24.1** (± 50.3)	**4.0** (± 1.4)	**5.2** (± 1.3)	0.0 (± 0.0)	N/A(± N/A)	N/A(± N/A)	N/A(± N/A)
BM8	0.0 (± 0.0)	N/A(± N/A)	N/A(± N/A)	N/A(± N/A)	**1.6** (± 0.9)	**63.7** (± 40.4)	**4.6** (± 0.9)	**4.8** (± 0.8)

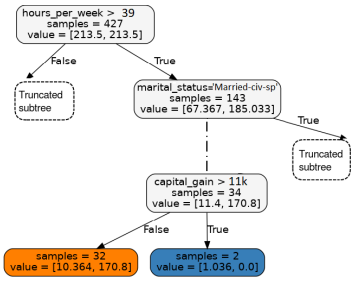
Answer RQ2: Simulated annealing outperforms other randomized search strategies in inding maximum k-discrimination, the time to the first instance of max k, and the number of unique instances with max k-discrimination.

Explanations via HYFAIR (RQ3). Our goal is to find out whether HYFAIR provides robust and succinct explanation compared to the baseline LIME [26]. Given a set of inputs that characterize the significant discrimination (a high value of K), we investigate the performance of HYFAIR (explanation) to LIME. We note that LIME cannot provide the explanation out-of-the-box since its goal is to explain the outcome for one sample. To enable LIME to provide such explanations, we query it with K samples and extract the common features from the top 3 features in each explanation. To study the robustness of LIME, we perturb the important features (one by one), measure the value of k-discrimination by querying the DNN model on the perturbed features, and report the difference between the initial and perturbed values.

We report the results of experiments in Table III, with the columns `Init.K` - the max k-discrimination value from the search step, `Size` - the size of explanation in terms of # of conjunction, `K` - the mean k-value before perturbation, `Pert.K` - the k-value after perturbation of the corresponding features for robustness, `Diff` - the difference between the initial and perturb explanations, and `Cov` - the input area volume of explanation for generality. Let us first study the overall trend in Table III. Considering the Adult census (AC) dataset, `Init.K` and `K` values vary between 17 to 20, `Size` value varies between 1 to 13, `Diff` value varies between 2.3 to 17.2, and `Cov` varies between 1 to 2,590.

Looking into Table III, HYFAIR has given an explanation both with smaller or equal size, but with more robust validation rules in terms of diff values, and with significantly higher coverage. There are very few cases when HYFAIR requires a larger size of explanation, but in those cases, the explanation rules show strong performances in terms of coverage and validation diff values. Specifically, HYFAIR performs better in more than 78% of the cases with respect to the robustness of explanations. In all cases, HYFAIR covers a larger volume of input space than LIME. HYFAIR provides a succinct explanation, whereas LIME's explanations may use all features, which may make it difficult to find the root causes.

Concrete Example of Explanation. HYFAIR also provides rich explanation through the hyper-rectangular split of the input space to show the significant discrimination (see all decision tree models in the appendix). The following plot shows a DT that explains the circumstances under which the DNN model for the AC2 benchmark becomes significantly discriminatory. For example, the predicate {hours_per_week $>$ 39 $\wedge$ marital_status != [‘Married-civ spouse’] $\wedge$ hours_per_week $<=$ 64 $\wedge$ workclass = [‘Private’, ‘Self-emp-not-inc’, ‘Self-emp-inc’, ‘Federal-gov’] $\wedge$ native_country = [‘United-States’, ‘Cambodia’, ‘England’, ‘Puerto-Rico’, ‘Canada’, ‘Germany’] $\wedge$ capital_gain $<=$ 11000} shows a hyper-rectangular region with a significant arbitrariness (note: a subset of conditions are not shown in the DT due to the presentation).



Answer RQ3: HYFAIR provides rich and succinct explanations that cover a larger sub-space and is more robust in explaining the root cause of significant arbitrariness.

Debiasing unfairness via HYFAIR (RQ4). We perform two interventions to mitigate unfairness in the original models: 1) we add decision tree rules as guardrails to refute queries that can lead to arbitrary outcomes, and 2) we retrain the original DNN models over curated discriminatory instances via data augmentation, similar to prior works [42], [43], [10],

TABLE II: Random Walk (RW) vs. Simulated Annealing (SA) as well as K-nearest neighbors variant of SA.

Model	Search	Iter(× 1K)	$Max.K$	$Avg.K$	$T.1st$	$\#ID$	$Succ.rate$	$\#ID.maxK.1$	$T.maxK.1$
AC1	RW	2.7 (± 0.4)	**20.0** (± 0.0)	8.1 (± 0.0)	5 (± 10)	1809 (± 239)	**67.2** (± 0.7)	**3.8** (± 2.1)	4561 (± 6387)
	SA	**3.6** (± 1.1)	**20.0** (± 0.0)	**9.3** (± 1.1)	**1** (± 2)	**1927** (± 553)	53.3 (± 1.8)	2.7 (± 1.4)	**1162** (± 1723)
	SA+KNN	2.1 (± 0.3)	19.3 (± 0.5)	8.8 (± 0.2)	62 (± 86)	625 (± 200)	29.7 (± 9.9)	1.9 (± 0.7)	4799 (± 4089)
AC2	RW	2.0 (± 0.4)	18.8 (± 0.4)	14.2 (± 0.0)	3 (± 3)	1953 (± 366)	**97.4** (± 0.2)	2.0 (± 2.2)	6761 (± 7330)
	SA	**2.6**(± 0.4)	**19.0** (± 0.0)	14.0(± 0.8)	**2** (± 1)	**2245** (± 312)	87.7 (± 0.6)	3.8 (± 2.3)	4769 (± 3939)
	SA+KNN	1.3(± 0.3)	18.3 (± 0.5)	**14.3** (± 0.1)	8 (± 2)	885(± 149)	71.7 (± 14.0)	**4.0** (± 2.5)	**3328** (± 3733)
AC3	RW	2.6(± 0.5)	**20.0** (± 0.0)	14.1 (± 0.1)	**4** (± 6)	1667 (± 310)	**64.3** (± 0.5)	**255.2** (± 51.7)	**63** (± 58)
	SA	**3.2** (± 0.5)	**20.0** (± 0.0)	14.1 (± 2.2)	**4** (± 7)	**1846** (± 353)	58.6 (± 6.3)	170.4 (± 72.0)	86 (± 80)
	SA+KNN	2.0 (± 0.5)	**20.0** (± 0.0)	**17.5** (± 0.5)	20 (± 20)	105(± 32)	5.8 (± 2.9)	12.5 (± 6.2)	808 (± 1504)
AC4	RW	**0.2** (± 0.0)	**20.0** (± 0.0)	15.2 (± 0.4)	**122** (± 66)	122 (± 4)	88.0 (± 3.3)	4.9 (± 1.7)	2369 (± 2335)
	SA	**0.2** (± 0.0)	**20.0** (± 0.0)	**17.1** (± 0.4)	127 (± 47)	**125**(± 7)	**91.0** (± 4.7)	**13.2** (± 4.9)	**1262** (± 642)
	SA+KNN	0.1(± 0.0)	19.7 (± 0.5)	16.2 (± 1.1)	229 (± 151)	54 (± 15)	72.6 (± 22.7)	4.7 (± 6.7)	2841 (± 2781)
AC5	RW	**1.7** (± 0.3)	18.1 (± 0.5)	**13.7** (± 0.0)	4 (± 3)	**1650** (± 333)	**97.9** (± 0.3)	1.8 (± 1.0)	4730 (± 6327)
	SA	1.6(± 0.2)	**18.3** (± 0.4)	13.0 (± 0.1)	**3** (± 1)	1545 (± 235)	97.7 (± 0.76)	**9.6** (± 5.7)	**2209** (± 2826)
	SA+KNN	0.9 (± 0.2)	17.3 (± 0.7)	13.5 (± 0.2)	11 (± 4)	687 (± 176)	78.4 (± 12.1)	6.0 (± 13.8)	6483 (± 4482)
AC6	RW	2.6 (± 0.5)	**20.0** (± 0.0)	14.3 (± 0.0)	**1** (± 0)	2491 (± 473)	**94.2** (± 0.3)	37.9 (± 6.2)	338 (± 230)
	SA	**2.9** (± 0.6)	**20.0** (± 0.0)	13.8 (± 0.3)	**1** (± 1)	**2623** (± 500)	91.5 (± 1.5)	**108.4** (± 25.7)	**93** (± 82)
	SA+KNN	1.8 (± 0.4)	**20.0** (± 0.0)	**14.6** (± 0.3)	6 (± 1)	801 (± 197)	45.4 (± 10.8)	12.8 (± 3.9)	1847 (± 2049)
AC7	RW	4.1 (± 0.8)	16.0 (± 0.0)	14.4 (± 0.3)	**820** (± 770)	17 (± 3)	**0.4** (± 0.1)	**7.1** (± 1.8)	**3493** (± 2993)
	SA	**4.3** (± 0.9)	**16.5** (± 0.5)	14.3 (± 0.4)	1602(± 1527)	**18** (± 7)	**0.4** (± 0.2)	2.6 (± 1.6)	3646 (± 2413)
	SA+KNN	3.8 (± 1.0)	16.0 (± 0.5)	**15.0** (± 1.0)	2019 (± 1579)	7 (± 3)	0.2 (± 0.1)	2.9 (± 1.8)	7189 (± 4688)
AC8	RW	2.5 (± 0.6)	16.0 (± 0.0)	11.2 (± 0.0)	11(± 13)	1467 (± 370)	**58.5** (± 0.8)	7.5 (± 3.3)	**1804** (± 1785)
	SA	**3.8** (± 1.5)	**16.9** (± 0.3)	**12.3** (± 0.2)	**1** (± 2)	**2072** (± 518)	57.0 (± 8.8)	**9.2** (± 3.6)	2569 (± 2182)
	SA+KNN	2.2(± 0.4)	15.6 (± 0.7)	12.0 (± 0.2)	12 (± 9)	222 (± 48)	10.3 (± 2.1)	5.0 (± 4.6)	3061 (± 3627)
AC9	RW	2.3 (± 0.4)	**19.0** (± 0.0)	**14.1** (± 0.0)	2 (± 3)	2080 (± 343)	**91.3** (± 2.9)	**53.3** (± 8.3)	**102** (± 41)
	SA	**3.1** (± 0.4)	**19.0** (± 0.0)	13.7 (± 0.7)	**1** (± 0)	**2430** (± 323)	77.6 (± 1.4)	51.0 (± 8.9)	443 (± 362)
	SA+KNN	2.3 (± 0.3)	**19.0** (± 0.0)	13.8 (± 0.2)	9 (± 6)	443 (± 183)	19.7 (± 9.7)	5.2 (± 1.5)	1263 (± 1258)
AC10	RW	2.3(± 0.4)	**17.0** (± 0.0)	11.9 (± 0.0)	**3** (± 8)	1795 (± 311)	**77.1** (± 0.5)	1.8 (± 0.9)	4245 (± 5628)
	SA	**3.5** (± 0.9)	16.9 (± 0.3)	11.3 (± 1.6)	**3** (± 4)	**1838** (± 545)	51.6 (± 2.1)	**5.8** (± 11.5)	**3778** (± 3017)
	SA+KNN	2.3 (± 0.3)	16.4 (± 0.5)	**13.0** (± 0.1)	15 (± 18)	336 (± 62)	14.8 (± 3.7)	4.5 (± 3.9)	4779 (± 4621)
AC11	RW	1.1 (± 0.3)	**20.0** (± 0.0)	13.8 (± 0.1)	10 (± 6)	89 (± 275)	**81.2** (± 1.3)	7.2 (± 3.1)	2990 (± 2735)
	SA	**1.3** (± 0.3)	**20.0** (± 0.0)	**15.0** (± 0.4)	**8** (± 5)	**987** (± 275)	73.8 (± 7.1)	**29.9** (± 16.2)	**471** (± 465)
	SA+KNN	0.6 (± 0.2)	19.4 (± 0.5)	14.3 (± 0.6)	44 (± 57)	155 (± 80)	26.1 (± 12.1)	5.1 (± 4.1)	2498 (± 2158)
AC12	RW	2.2(± 0.4)	**20.0** (± 0.0)	**17.7** (± 0.1)	**1** (± 1)	**2204** (± 409)	**99.2** (± 0.2)	**99.7** (± 24.8)	**120** (± 26)
	SA	**2.4** (± 0.9)	**20.0** (± 0.0)	15.7 (± 1.7)	**1** (± 0)	1662 (± 593)	70.8 (± 1.6)	60.6 (± 25.6)	203 (± 149)
	SA+KNN	1.5 (± 0.1)	**20.0** (± 0.0)	**17.7** (± 0.1)	9 (± 4)	957 (± 185)	65.5 (± 11.0)	19.6 (± 8.0)	2068 (± 1431)
BM1	RW	8.8 (± 1.9)	**9.0** (± 0.0)	5.0 (± 0.0)	3 (± 4)	**5409** (± 1151)	**61.2** (± 0.3)	**45.8** (± 10.2)	**295** (± 43)
	SA	**9.1** (± 1.6)	**9.0** (± 0.0)	5.2 (± 0.6)	**1** (± 1)	5135 (± 657)	57.3 (± 6.1)	26.3 (± 8.5)	420 (± 393)
	SA+KNN	4.7(± 0.9)	**9.0** (± 0.0)	**5.7** (± 0.1)	9 (± 11)	1663 (± 387)	35.3 (± 3.8)	13.4(± 5.4)	1646 (± 1018)
BM2	RW	11.0 (± 0.8)	**9.0** (± 0.0)	4.0 (± 0.0)	12 (± 13)	**766** (± 53)	**7.0** (± 0.1)	**22.2** (± 2.0)	**738** (± 682)
	SA	**11.3** (± 0.0)	**9.0** (± 0.0)	4.6 (± 1.1)	**8** (± 10)	726 (± 200)	6.4 (± 1.8)	17.3 (± 7.0)	936 (± 896)
	SA+KNN	9.0 (± 1.8)	**9.0** (± 0.0)	**5.5** (± 0.1)	32 (± 28)	417 (± 94)	4.6 (± 0.2)	7.9 (± 3.3)	3248 (± 3087)
BM3	RW	10.6 (± 1.8)	**9.0** (± 0.0)	6.0 (± 0.0)	3 (± 3)	**3230** (± 551)	**30.3** (± 0.0)	341.4 (± 57.1)	79 (± 56)
	SA	**11.3** (± 0.1)	**9.0** (± 0.0)	6.5 (± 0.9)	**2** (± 2)	2984 (± 315)	26.5 (± 2.9)	**419.1** (± 132.0)	**24** (± 17)
	SA+KNN	7.9 (± 2.2)	**9.0** (± 0.0)	**7.3** (± 0.1)	21 (± 36)	1072 (± 276)	13.6 (± 0.7)	111.1 (± 40.3)	131 (± 148)
BM5	RW	**11.0** (± 0.9)	**6.0** (± 0.0)	3.6 (± 0.0)	**2** (± 2)	**2101** (± 182)	**19.2** (± 0.1)	5.9 (± 0.3)	1620 (± 1549)
	SA	10.2 (± 1.7)	**6.0** (± 0.0)	3.7 (± 0.5)	3 (± 3)	1605 (± 249)	15.9 (± 2.5)	25.4 (± 22.2)	**464** (± 2056)
	SA+KNN	10.7 (± 1.6)	5.9 (± 0.3)	**4.1** (± 0.0)	11 (± 9)	1366 (± 213)	12.8 (± 0.4)	**29.2** (± 81.1)	4836 (± 2269)
BM6	RW	**11.3** (± 0.0)	7.0 (± 0.0)	2.2 (± 0.0)	1.7 (± 1.7)	**2450** (± 0)	**21.7** (± 0.0)	4.0 (± 0.0)	1002 (± 732)
	SA	**11.3** (± 0.0)	**8.0** (± 0.0)	**3.4** (± 0.0)	**1.6** (± 1.5)	2315 (± 328)	20.5 (± 2.9)	**21.5** (± 13.1)	**137** (± 32)
	SA+KNN	**11.3** (± 0.1)	7.9 (± 0.3)	2.8 (± 0.1)	5.0 (± 4.1)	2123 (± 238)	18.8 (± 2.1)	3.4 (± 3.6)	3239 (± 1900)
BM8	RW	**0.3** (± 0.1)	7.0 (± 0.0)	3.8 (± 0.1)	**129** (± 71)	**52** (± 17)	**18.5** (± 2.6)	**1.2** (± 0.5)	4977 (± 5861)
	SA	**0.3** (± 0.1)	**8.0** (± 0.0)	**4.8** (± 0.1)	394 (± 87)	42 (± 4)	16.5 (± 3.5)	**1.2** (± 2.1)	**4570** (± 5023)
	SA+KNN	0.1 (± 0.0)	6.5 (± 1.7)	4.3 (± 0.3)	3415 (± 478)	20 (± 9)	12.7 (± 4.0)	2.0 (± 2.0)	8929.7 (± 4458)

TABLE III: LIME [26] vs. HYFAIR based on robustness, size, and the generality of explanation.

DNN	$Init.K$	LIME [26]					HYFAIR				
		$Size$	K	$Pert.K$	$Diff$	Cov	$Size$	K	$Pert.K$	$Diff$	Cov
AC1	20	5	20	8.5 (± 3.3)	11.5	1	6	12.7 (± 6.2)	1.1 (± 0.3)	**11.6**	**108.2 (±8.0)**
AC2	19	6	19	9.9 (± 3.4)	9.1	1	7	11.7 (± 3.0)	2.5 (± 2.5)	**9.2**	**15.4 (±3.7)**
AC3	20	5	20	11 (± 4.3)	9.8	1	5	15.8 (± 2.0)	4.3 (± 4.5)	**11.5**	**12.0 (±2.2)**
AC4	20	7	20	16.1 (± 3)	3.9	1	**3**	19.5 (± 0.7)	8.2 (± 5.9)	**11.3**	**384.7 (±23.7)**
AC5	18	4	18	11 (± 0.7)	7.0	1	**3**	14.0 (± 0.0)	6.9 (± 3.2)	**7.1**	**1572.0 (±34.9)**
AC6	20	4	20	17.9 (± 0.8)	2.3	1	**3**	17.0 (± 0.0)	8.2 (± 2.6)	**8.8**	**622.3 (± 22.8)**
AC7	17	13	17	12 (± 3.6)	5.0	1	**3**	17.0 (± 0.0)	1.0 (± 0.0)	**16.0**	1.0 (± 0.0)
AC8	17	5	17	9.4 (± 1.4)	7.6	1	6	11.2 (± 0.9)	1.2 (± 0.6)	**10.0**	**57.4 (± 8.9)**
AC9	19	**3**	19	6.7 (± 1.7)	**12.3**	1	4	16.0 (± 0.0)	4.0(± 0.0)	12.0	**493.0 (± 25.8)**
AC10	17	5	17	7.9 (± 4.3)	**9.1**	1	3	17.0 (± 0.0)	10.0 (± 0.0)	7.0	**2590.5 (± 55.4)**
AC11	20	6	20	16 (± 1.7)	4.2	1	**4**	13.0 (± 1.4)	4.3 (± 2.1)	**8.7**	**895.3 (± 27.5)**
AC12	20	6	20	10.8 (± 2.7)	9.2	1	**4**	19.0 (± 1.4)	1.8(± 1.2)	**17.2**	**702.9 (± 21.7)**
BM1	9	3	9	3.7 (± 1.0)	5.3	1	4	9.0 (± 0.0)	1.9 (± 2.2)	**7.1**	**77.5 (± 7.9)**
BM2	9	4	9	5.3 (± 0.6)	3.7	1	8	5.2 (± 3.2)	1.2 (± 0.6)	**4.0**	**19.1 (± 4.6)**
BM3	9	3	9	5.7 (± 0.0)	3.3	1	4	8.0 (± 0.6)	2.8 (± 1.1)	**5.2**	1.0 (± 0.0)
BM5	6	**3**	6	2.4 (± 0.0)	**3.6**	1	7	5.0 (± 0.0)	1.9 (± 1.1)	3.1	**1.0 (± 1.2)**
BM6	8	3	8	3.0 (± 0.0)	2.0	1	4	6.7 (± 1.2)	2.5 (± 0.5)	**4.2**	**88.5 (±7.5)**
BM8	8	4	8	3.4 (± 0)	**4.6**	1	6	8.0 (± 0)	4.1 (± 1.1)	3.9	1.0 (± 0.0)

TABLE IV: RQ4: Original vs. Debiased model (with/without decision tree rules).

DNN	Intervention	Acc (%)	Iter(× 1K)	$Max.K$	$Avg.K$	$\#ID$	$Succ.rate$ (%)	$\#ID.maxK.1$
AC1	Original	81.81	3.6 (± 1.1)	20.0 (± 0.0)	9.3 (± 1.1)	1927.0 (± 552.5)	53.3 (± 1.8)	2.8 (± 1.4)
	Original w DT	81.81	3.7 (± 1.2)	**18.1** (± 0.3)	**9.1** (± 1.0)	2316.0 (± 357.4)	68.6 (± 23.2)	17.5 (± 8.8)
	Debias w/o DT	81.22	3.6 (± 0.0)	20.0 (± 0.0)	13.2 (± 0.1)	1172.3 (± 9.2)	32.5 (± 0.2)	1.7 (± 0.6)
	Debias w DT	81.22	3.6 (± 0.0)	19.5 (± 0.8)	13.6 (± 0.4)	**1112.7** (± 605.7)	**30.9** (± 16.8)	4.6 (± 5.0)
AC2	Original	83.19	2.6 (± 0.4)	19.0 (± 0.0)	14.0 (± 0.8)	2245.0 (± 318.8)	87.7 (± 0.9)	3.8 (± 2.2)
	Original w DT	83.19	2.5 (± 0.4)	**18.2** (± 0.4)	14.2 (± 0.1)	955.0 (± 131.9)	37.6 (± 2.1)	**1.5** (± 0.7)
	Debias w/o DT	82.32	2.6 (± 0.0)	20.0 (± 0.0)	**4.8** (± 0.0)	1244.0 (± 0.0)	48.5 (± 0.0)	3.0 (± 0.0)
	Debias w DT	82.32	2.2 (± 0.5)	19.5 (± 0.5)	5.2 (± 0.5)	**836.0** (± 304.1)	37.8 (± 10.1)	4.2 (± 1.7)
AC3	Original	83.29	3.2 (± 0.9)	20.0 (± 0.0)	14.3 (± 2.2)	1794.1 (± 380.5)	58.0 (± 6.3)	166.8 (± 69.8)
	Original w DT	83.29	3.2 (± 0.9)	**19.7** (± 0.5)	**11.9** (± 3.9)	**82.1** (± 36.8)	**2.8** (± 1.6)	**2.5** (± 1.1)
	Debias w/o DT	82.25	3.2 (± 0.0)	20.0 (± 0.0)	16.8 (± 0.0)	2230.0 (± 0.0)	70.5 (± 0.0)	490.0 (± 0.0)
	Debias w DT	82.25	3.2 (± 0.0)	20.0 (± 0.0)	16.8 (± 0.4)	1890.7 (± 556.4)	59.7 (± 17.6)	422.8 (± 111.7)
AC4	Original	82.74	0.1 (± 0.0)	20.0 (± 0.0)	17.1 (± 0.4)	124.8 (± 7.3)	90.5 (± 4.6)	13.2 (± 4.9)
	Original w DT	82.74	0.1 (± 0.0)	16.9 (± 2.2)	12.9 (± 2.9)	3.8 (± 2.1)	2.8 (± 1.5)	0.9 (± 0.3)
	Debias w/o DT	82.60	0.1 (± 0.0)	**5.0** (± 10.0)	**5.0** (± 10.0)	**0.2** (± 0.5)	**0.2** (± 0.4)	**0.2** (± 0.5)
	Debias w DT	82.60	0.1 (± 0.0)	11.8 (± 10.8)	9.2 (± 8.4)	18.5 (± 21.9)	13.2 (± 15.5)	2.7 (± 3.4)
AC5	Original	83.36	1.6 (± 0.2)	18.2 (± 0.5)	13.0 (± 0.1)	1545.2 (± 235.1)	97.7 (± 0.8)	9.6 (± 5.7)
	Original w DT	83.36	1.6 (± 0.2)	**17.5** (± 0.7)	13.5 (± 0.1)	866.5 (± 141.3)	55.1 (± 1.4)	**2.4** (± 2.0)
	Debias w/o DT	82.66	1.3 (± 0.4)	18.5 (± 2.1)	11.1 (± 0.9)	696.0 (± 79.2)	57.3 (± 13.9)	15.5 (± 17.7)
	Debias w DT	82.66	1.4 (± 0.3)	19.5 (± 1.7)	**10.7** (± 0.8)	**684.8** (± 79.6)	**49.2** (± 7.3)	15.2 (± 14.2)
AC6	Original	82.04	2.9 (± 0.5)	20.0 (± 0.0)	13.8 (± 0.3)	2623.2 (± 499.7)	91.5 (± 1.5)	108.4 (± 25.7)
	Original w DT	82.04	2.8 (± 0.6)	**18.8** (± 0.9)	13.6 (± 0.3)	**208.7** (± 38.7)	**7.4** (± 0.6)	**2.9** (± 3.7)
	Debias w/o DT	80.86	2.9 (± 0.0)	20.0 (± 0.0)	10.3 (± 0.0)	312.0 (± 0.0)	10.9 (± 0.0)	2.0 (± 0.0)
	Debias w DT	80.86	2.9 (± 0.0)	19.2 (± 1.2)	**10.0** (± 0.2)	285.7 (± 27.6)	10.0 (± 1.0)	2.0 (± 1.1)
AC7	Original	82.96	4.3 (± 0.9)	16.5 (± 0.5)	14.3 (± 0.4)	17.9 (± 6.7)	0.4 (± 0.1)	2.6 (± 1.6)
	Original w DT	82.96	4.3 (± 0.9)	11.0 (± 0.0)	10.2 (± 1.1)	0.3 (± 0.6)	0.0 (± 0.0)	0.2 (± 0.4)
	Debias w/o DT	81.91	3.2 (± 0.0)	**0.0** (± 0.0)	**0.0** (± 0.0)	**0.0** (± 0.0)	**0.0** (± 0.0)	**0.0** (± 0.0)
	Debias w DT	81.91	2.3 (± 1.3)	9.5 (± 13.4)	8.3 (± 11.7)	9.0 (± 12.7)	0.6 (± 0.9)	2.0 (± 2.8)
AC8	Original	82.45	3.8 (± 1.5)	16.9 (± 0.3)	12.3 (± 0.2)	2071.6 (± 517.7)	57.0 (± 8.8)	9.2 (± 3.6)
	Original w DT	82.45	3.9 (± 1.5)	**16.0** (± 0.0)	**11.7** (± 0.6)	**856.9** (± 160.3)	**24.1** (± 7.6)	**5.2** (± 2.6)
	Debias w/o DT	81.65	3.8 (± 0.0)	20.0 (± 0.0)	18.3 (± 0.1)	3088.5 (± 20.5)	81.3 (± 0.6)	1255.0 (± 25.5)
	Debias w DT	81.65	3.8 (± 0.0)	20.0 (± 0.0)	18.4 (± 0.1)	3135.0 (± 1.4)	82.4 (± 0.1)	1338.5 (± 113.8)
AC9	Original	82.07	3.1 (± 0.4)	19.0 (± 0.0)	13.7 (± 0.7)	2430.4 (± 323.3)	77.6 (± 1.4)	51.0 (± 8.9)
	Original w DT	82.07	3.2 (± 0.5)	**18.0** (± 0.0)	**13.5** (± 1.2)	**102.6** (± 20.3)	**3.3** (± 0.8)	**7.6** (± 2.0)
	Debias w/o DT	80.96	3.1 (± 0.0)	19.0 (± 0.0)	16.4 (± 0.0)	2766.0 (± 0.0)	88.2 (± 0.0)	25.0 (± 0.0)
	Debias w DT	80.96	3.1 (± 0.0)	18.2 (± 1.5)	16.0 (± 0.5)	2051.8 (± 1367.1)	65.4 (± 43.6)	14.2 (± 9.9)
AC10	Original	81.68	3.5 (± 0.9)	16.9 (± 0.3)	**11.3** (± 1.6)	1838.2 (± 544.9)	51.6 (± 2.1)	5.8 (± 11.5)
	Original w DT	81.68	3.6 (± 1.0)	**16.0** (± 0.0)	13.0 (± 0.3)	**804.2** (± 279.5)	**22.0** (± 2.1)	**8.5** (± 2.8)
	Debias w/o DT	81.28	3.5 (± 0.0)	19.0 (± 0.0)	12.5 (± 0.0)	2101.0 (± 0.0)	59.4 (± 0.0)	104.0 (± 0.0)
	Debias w DT	81.28	3.5 (± 0.0)	19.0 (± 0.0)	12.5 (± 0.1)	2145.2 (± 106.5)	60.6 (± 3.0)	99.2 (± 9.2)
AC11	Original	82.19	1.3 (± 0.3)	20.0 (± 0.0)	15.0 (± 0.4)	987.3 (± 274.9)	73.8 (± 7.1)	29.9 (± 16.2)
	Original w DT	81.19	1.3 (± 0.3)	19.6 (± 0.5)	14.0 (± 0.8)	**435.1** (± 123.2)	**32.7** (± 2.6)	9.6 (± 11.9)
	Debias w/o DT	80.85	1.3 (± 0.0)	20.0 (± 0.0)	10.2 (± 0.0)	767.0 (± 0.0)	57.5 (± 0.0)	6.0 (± 0.0)
	Debias w DT	80.85	1.3 (± 0.0)	**17.7** (± 4.9)	**9.2** (± 1.5)	613.7 (± 521.2)	46.0 (± 39.1)	**3.3** (± 2.5)
AC12	Original	82.16	2.4 (± 0.9)	20.0 (± 0.0)	15.7 (± 1.7)	1661.8 (± 593.3)	70.8 (± 1.6)	60.6 (± 25.6)
	Original w DT	82.16	2.4 (± 0.9)	19.2 (± 0.6)	17.7 (± 0.6)	**6.8** (± 2.7)	**0.3** (± 0.1)	1.8 (± 1.5)
	Debias w/o DT	81.66	2.4 (± 0.0)	15.0 (± 0.0)	**9.2** (± 0.0)	1483.0 (± 0.0)	62.8 (± 0.0)	2.0 (± 0.0)
	Debias w DT	81.66	2.4 (± 0.0)	**14.2** (± 0.5)	**9.2** (± 0.1)	1111.5 (± 718.5)	47.1 (± 30.4)	9.8 (± 9.8)
BM1	Original	90.63	9.5 (± 1.5)	9.0 (± 0.0)	5.2 (± 0.6)	5357.6 (± 391.0)	57.2 (± 6.2)	28.3 (± 7.9)
	Original w DT	90.63	10.0 (± 1.5)	**5.8** (± 0.4)	**4.7** (± 0.4)	**21.8** (± 4.8)	**0.2** (± 0.1)	**2.6** (± 2.9)
	Debias w/o DT	90.11	5.9 (± 0.0)	9.0 (± 0.0)	6.1 (± 0.0)	335.5 (± 57.3)	5.6 (± 0.9)	26.0 (± 7.1)
	Debias w DT	90.11	4.1 (± 0.7)	8.5 (± 0.6)	5.8 (± 0.4)	136.8 (± 116.6)	3.5 (± 3.0)	10.8 (± 6.6)
BM2	Original	90.27	11.3 (± 0.0)	9.0 (± 0.0)	4.6 (± 1.1)	730.0 (± 195.9)	6.5 (± 1.7)	18.4 (± 5.8)
	Original w DT	90.63	11.3 (± 0.0)	8.4 (± 0.5)	5.1 (± 0.8)	**727.2** (± 760.7)	**6.4** (± 6.7)	5.5 (± 8.7)
	Debias w/o DT	89.86	11.3 (± 0.0)	9.0 (± 0.0)	6.0 (± 0.0)	2168.5 (± 34.6)	19.2 (± 0.3)	53.5 (± 0.7)
	Debias w DT	89.86	7.2 (± 5.8)	**4.5** (± 6.4)	**2.9** (± 4.2)	750.5 (± 1061.4)	6.6 (± 9.4)	9.0 (± 12.7)
BM3	Original	90.35	11.3 (± 0.1)	9.0 (± 0.0)	6.5 (± 0.9)	2988.4 (± 311.6)	26.5 (± 2.9)	422.8 (± 128.3)
	Original w DT	90.35	11.3 (± 0.1)	**7.2** (± 0.4)	**5.3** (± 0.9)	**12.9** (± 9.9)	**0.1** (± 0.1)	**3.5** (± 2.5)
	Debias w/o DT	90.17	6.7 (± 1.0)	9.0 (± 0.0)	7.8 (± 0.0)	1296.5 (± 207.2)	19.2 (± 0.2)	386.0 (± 91.9)
	Debias w DT	90.17	5.4 (± 1.3)	9.0 (± 0.0)	7.5 (± 0.0)	326.0 (± 387.2)	5.5 (± 6.4)	76.2 (± 90.3)
BM5	Original	90.26	10.2 (± 1.7)	6.0 (± 0.0)	3.7 (± 0.5)	1608.0 (± 248.6)	16.0 (± 2.4)	25.1 (± 22.4)
	Original w DT	90.26	11.3 (± 0.0)	**5.0** (± 0.0)	4.1 (± 0.0)	520.0 (± 17.9)	4.6 (± 0.2)	150.0 (± 19.7)
	Debias w/o DT	89.81	10.2 (± 0.0)	9.0 (± 0.0)	4.7 (± 0.0)	264.0 (± 26.9)	2.6 (± 0.3)	2.0 (± 0.0)
	Debias w DT	89.81	10.2 (± 0.0)	6.5 (± 4.4)	**3.5** (± 2.3)	**182.2** (± 121.6)	**1.8** (± 1.2)	3.8 (± 4.9)
BM6	Original	89.89	11.3 (± 0.0)	8.0 (± 0.0)	**3.4** (± 0.0)	2315.0 (± 328.3)	20.5 (± 2.9)	**21.5** (± 13.1)
	Original w DT	89.89	11.3 (± 0.0)	**6.0** (± 0.0)	2.6 (± 0.0)	**608.2** (± 38.1)	**5.4** (± 0.3)	26.8 (± 4.7)
	Debias w/o DT	89.03	11.3 (± 0.0)	9.0 (± 0.0)	6.1 (± 0.0)	2902.5 (± 16.3)	25.7 (± 0.1)	56.0 (± 2.8)
	Debias w DT	89.03	11.3 (± 0.0)	8.8 (± 0.5)	6.4 (± 0.4)	1963.2 (± 1346.0)	17.4 (± 11.9)	32.8 (± 21.9)
BM8	Original	90.07	0.3 (± 4.7)	8.0 (± 0.0)	4.8 (± 0.1)	41.8 (± 4.4)	16.5 (± 3.5)	4.2 (± 1.5)
	Original w DT	90.07	0.3 (± 0.1)	5.0 (± 0.0)	3.9 (± 0.2)	26.2 (± 4.0)	10.2 (± 1.6)	4.5 (± 1.7)
	Debias w/o DT	90.22	3.8 (± 3.0)	3.0 (± 4.2)	3.0 (± 4.2)	0.5 (± 0.7)	0.0 (± 0.0)	0.5 (± 0.7)
	Debias w DT	90.22	5.6 (± 2.0)	**0.0** (± 0.0)	**0.0** (± 0.1)	**0.1** (± 0.1)	**0.0** (± 0.0)	**0.2** (± 0.2)

[9] to obtain debiased models. Table IV shows the comparison between the original and debiased models with and without decision tree rules. Overall, the debiased models led to at most 2% reduction in the accuracy. We apply simulated annealing (SA) to search for unfairness in the mitigated models, similar to the original ones. We find that adding the decision tree rules to the original models outperform other techniques in reducing the maximum k-discrimination in 67% of cases. The debiased models with and without DT achieved better results in 22% and 11% of cases, respectively. Similarly, adding DT rules to the original models reduce the success rates of finding individual discrimination cases in 67% of cases where the debiased models with DT achieved better results in 28% of cases. When considering the average k-discrimination of ID samples, both original and debiased models with DT tie by reducing it in 39% of cases.

One interesting and unexpected outcome is that while the debiased models reduce the $Succ.rate$ and $\#ID$, they often increase $Max.K$ value. This shows that reducing k-discrimination with simple retraining strategies does not work and requires careful retraining and novel strategies. While adding decision tree rules help, we believe that retraining introduces new fairness vulnerabilities that require further iteration of HYFAIR to infer new discriminatory rules. We believe that more in-depth future research is necessary for debiasing k-discrimination bugs in the models.

Answer RQ4: Applying decision rules as guardrails for denying output in more sensitive cases with the original model (67%) and retrained models (22%) reduces k-discrimination metrics in 89% of cases.

VI. DISCUSSION

Limitation. In this work for generating counterfactuals, we perturb for all possible combinations of the protected attribute which might lead to some unrealistic or imaginary counterfactual instances. We use some rule-of-thumb relationships to mitigate this issue (e.g., a married individual with a female gender cannot be husband for the relationship attribute), but Conditional GANs and Variational Auto Encoders can be employed to improve the realism of samples [44].

Our current technique also does not handle intersectional fairness [45], [46], [47], which reveals unfairness in the combination of multiple protected attributes. To overcome this limitation, our proposed approach can be repeated multiple times (one per each combination) to certify fairness or find the maximum unfairness.

Threat to Validity. To ensure the validity of our experiments and the reproducibility and valid conclusion, we follow established rules and guidelines and take the average of the repeated experiments to validate our claims. To ensure that our results are generalizable and address external validity, and apply to multiple datasets, we experiment on 20 DNN models taken from the literature of fairness testing and the real world in Kaggle and use 2 different datasets. Our certification is limited to a given fairness notion, bounded to a time-out, and sensitive to seed selection. Hence, we may not guarantee fairness in general. Decision tree algorithms have the limitation of hyper-rectangular partitioning and may not show the causal relationships between input features and discriminatory instances.

VII. RELATED WORK

Verifying Fairness Properties. Multiple prior works used formal techniques to certify fairness in the ML models [48], [18], [7], [6], [49], [50]. FAIRIFY [6] addressed the fairness verification problem of neural networks for individual fairness. They formulate pre-trained DNNs via Satisfiability modulo theories (SMT), and either certify the DNN for individual fairness or find a counterexample that is an instance of individual discrimination. However, these approaches cannot distinguish between counterexamples, which are critical for prioritizing counterexamples and explaining patterns in fairness bugs.

Testing Fairness Properties. Multiple research works [30], [8], [51], [32], [52] consider testing the individual discrimination in non-neuron ML models. THEMIS [30], AEQUITAS [9], ADF [10], AFT [42], EXPGA [43], NEURONFAIR [11], and EIDIG [53] used causal fairness definition (different variants of 2-fairness notions) that may not prioritize test cases and quantify different risks of harm. DICE [20] employs an information theory-based method to quantify individual discrimination. However, DICE cannot guarantee the absence of unfairness. We use formal techniques to certify fairness and explain the root cause of bugs.

eXplainable AI. PARFAIT-ML [54] used decision trees to explain what hyperparameter configuration of ML libraries can lead to inferring unfair ML models. Mothilal et al. [55] provided diverse counterfactual explanations for a given decision subject that enables them to flip an ML decision outcome. Watcher et al. [56] focused on understanding the decision flip by the feature-perturbed version of the same individual. LORE [57] used a decision tree to approximate the non-linear models, whereas ANCHORS [58] used model-agnostic explanations based on if-then rules. HYFAIR is geared towards DNN software, and it goes beyond i) explanation of decision for one subject and ii) prevalent differential analysis.

VIII. CONCLUSION

This paper presented a hybrid framework for fairness analysis of neural networks by combining testing and verification techniques. We introduced a quantitative generalization of individual discrimination and proposed a method to explain the conditions under which DNN models exhibit significant clustered discrimination. Our approach supports both the detection and mitigation of such fairness violations. An important direction for future work is to systematically assess the risk posed by automated decision-support systems when a small number of marginalized groups receive unfavorable outcomes, even as the system appears fair to the majority.

Acknowledgment. The authors thank ASE reviewers for their time and invaluable feedback to improve this work. This project has been supported by NSF under Grant No. CNS-2230060, CNS-2527657, CNS-2230061, and CCF-2317207.

REFERENCES

[1] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. http://www.deeplearningbook.org.

[2] R. Berk and J. Bleich, "Forecasts of violence to inform sentencing decisions," *Journal of Quantitative Criminology*, vol. 30, pp. 79–96, 2014.

[3] G. Siocon, "Ways ai is changing hr departments," 2023.

[4] R. Gusmano, "How ai is adding faster funding and efficiency to small-business lending," 2024.

[5] J. Buolamwini and T. Gebru, "Gender shades: Intersectional accuracy disparities in commercial gender classification," in *Conference on fairness, accountability and transparency*, pp. 77–91, PMLR, 2018.

[6] S. Biswas and H. Rajan, "Fairify: Fairness verification of neural networks," in *Proceedings of the 45th International Conference on Software Engineering*, ICSE '23, p. 1546–1558, IEEE Press, 2023.

[7] H. Khedr and Y. Shoukry, "Certifair: A framework for certified global fairness of neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, pp. 8237–8245, 2023.

[8] 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.

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

[10] 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*, pp. 949–960, 2020.

[11] 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)*, pp. 1519–1531, 2022.

[12] 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*, ESEC/FSE 2021, (New York, NY, USA), p. 429–440, Association for Computing Machinery, 2021.

[13] Z. Chen, J. M. Zhang, F. Sarro, and M. Harman, "Maat: a novel ensemble approach to addressing fairness and performance bugs for machine learning software," in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2022, (New York, NY, USA), p. 1122–1134, Association for Computing Machinery, 2022.

[14] K. Peng, J. Chakraborty, and T. Menzies, "Fairmask: Better fairness via model-based rebalancing of protected attributes," 2022.

[15] S. Barocas, M. Hardt, and A. Narayanan, *Fairness and Machine Learning: Limitations and Opportunities*. MIT Press, 2023.

[16] K. Creel and D. Hellman, "The algorithmic leviathan: Arbitrariness, fairness, and opportunity in algorithmic decision-making systems," *Canadian Journal of Philosophy*, vol. 52, no. 1, pp. 26–43, 2022.

[17] 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*, pp. 214–226, 2012.

[18] A. Albarghouthi, L. D'Antoni, S. Drews, and A. V. Nori, "Fairsquare: probabilistic verification of program fairness," *Proceedings of the ACM on Programming Languages*, vol. 1, no. OOPSLA, pp. 1–30, 2017.

[19] M. J. Kusner, J. Loftus, C. Russell, and R. Silva, "Counterfactual fairness," *Advances in neural information processing systems*, vol. 30, 2017.

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

[21] Y. Li, J. Wang, and C. Wang, "Certifying the fairness of KNN in the presence of dataset bias," in *Computer Aided Verification - 35th International Conference, CAV 2023, Paris, France, July 17-22, 2023, Proceedings, Part II* (C. Enea and A. Lal, eds.), vol. 13965, pp. 335–357, Springer, 2023.

[22] 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* (S. Shoham and Y. Vizel, eds.), vol. 13372 of *Lecture Notes in Computer Science*, pp. 364–385, Springer, 2022.

[23] M. Fischetti and J. Jo, "Deep neural networks and mixed integer linear optimization," *Constraints*, vol. 23, no. 3, pp. 296–309, 2018.

[24] A. Kampmann, N. Havrikov, E. O. Soremekun, and A. Zeller, "When does my program do this? learning circumstances of software behavior," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2020, (New York, NY, USA), p. 1228–1239, Association for Computing Machinery, 2020.

[25] K. Gaaloul, C. Menghi, S. Nejati, L. C. Briand, and D. Wolfe, "Mining assumptions for software components using machine learning," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 159–171, 2020.

[26] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, (New York, NY, USA), pp. 1135–1144, Association for Computing Machinery, 2016.

[27] "EthicalML-XAI: An explainability toolbox for machine learning." https://github.com/EthicalML/xai, 2024. online.

[28] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[29] N. Yu, L. Carreon, G. Tan, and S. Tizpaz-Niari, "Fairlay-ml: Intuitive debugging of fairness in data-driven social-critical software," in *2025 IEEE/ACM 47th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pp. 25–28, IEEE, 2025.

[30] 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*, pp. 871–875, 2018.

[31] D. Dua and C. Graff, "UCI machine learning repository," 2017.

[32] 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*, ICSE '22, (New York, NY, USA), p. 871–882, Association for Computing Machinery, 2022.

[33] D. Gopinath, G. Katz, C. S. Păsăreanu, and C. Barrett, "Deepsafe: A data-driven approach for assessing robustness of neural networks," in *Automated Technology for Verification and Analysis: 16th International Symposium, ATVA 2018, Los Angeles, CA, USA, October 7-10, 2018, Proceedings 16*, pp. 3–19, Springer, 2018.

[34] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Towards proving the adversarial robustness of deep neural networks," *arXiv preprint arXiv:1709.02802*, 2017.

[35] N. Narodytska, S. Kasiviswanathan, L. Ryzhyk, M. Sagiv, and T. Walsh, "Verifying properties of binarized deep neural networks," in *AAAI'18*, 2018.

[36] S. Dutta, S. Jha, S. Sankaranarayanan, and A. Tiwari, "Output range analysis for deep feedforward neural networks," in *NASA Formal Methods Symposium*, pp. 121–138, Springer, 2018.

[37] V. A. Dasu, A. Kumar, S. Tizpaz-Niari, and G. Tan, "Neufair: Neural network fairness repair with dropout," in *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pp. 1541–1553, 2024.

[38] S. Biswas and H. Rajan, "Do the machine learning models on a crowd sourced platform exhibit bias? an empirical study on model fairness," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2020, (New York, NY, USA), p. 642–653, Association for Computing Machinery, 2020.

[39] 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*, ICSE '20, (New York, NY, USA), p. 949–960, Association for Computing Machinery, 2020.

[40] C. Urban, M. Christakis, V. Wüstholz, and F. Zhang, "Perfectly parallel fairness certification of neural networks," *Proc. ACM Program. Lang.*, vol. 4, nov 2020.

[41] D. Mazzucato and C. Urban, "Reduced products of abstract domains for fairness certification of neural networks," in *Static Analysis: 28th*

International Symposium, SAS 2021, Chicago, IL, USA, October 17–19, 2021, Proceedings, (Berlin, Heidelberg), p. 308–322, Springer-Verlag, 2021.

[42] Z. Zhao, T. Toda, and T. Kitamura, "Approximation-guided fairness testing through discriminatory space analysis," in *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, ASE '24, (New York, NY, USA), p. 1007–1018, Association for Computing Machinery, 2024.

[43] 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*, ICSE '22, (New York, NY, USA), p. 871–882, Association for Computing Machinery, 2022.

[44] Y. Xiao, A. Liu, T. Li, and X. Liu, "Latent imitator: Generating natural individual discriminatory instances for black-box fairness testing," in *Proceedings of the 32nd ACM SIGSOFT international symposium on software testing and analysis*, pp. 829–841, 2023.

[45] M. Zhang and J. Sun, "Adaptive fairness improvement based on causality analysis," in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2022, (New York, NY, USA), p. 6–17, Association for Computing Machinery, 2022.

[46] Z. Chen, J. M. Zhang, F. Sarro, and M. Harman, "Fairness improvement with multiple protected attributes: How far are we?," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ICSE '24, (New York, NY, USA), Association for Computing Machinery, 2024.

[47] A. Ghosh, L. Genuit, and M. Reagan, "Characterizing intersectional group fairness with worst-case comparisons," in *Proceedings of 2nd Workshop on Diversity in Artificial Intelligence (AIDBEI)* (D. Lamba and W. H. Hsu, eds.), vol. 142 of *Proceedings of Machine Learning Research*, pp. 22–34, PMLR, 09 Feb 2021.

[48] P. G. John, D. Vijaykeerthy, and D. Saha, "Verifying individual fairness in machine learning models," in *Conference on Uncertainty in Artificial Intelligence*, pp. 749–758, PMLR, 2020.

[49] B. H. Kim, J. Wang, and C. Wang, "Fairquant: Certifying and quantifying fairness of deep neural networks," 2024.

[50] V. Monjezi, A. Kumar, G. Tan, A. Trivedi, and S. Tizpaz-Niari, "Causal graph fuzzing for fair ml sofware development," in *Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings*, pp. 402–403, 2024.

[51] 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*, ESEC/FSE 2019, p. 625–635, 2019.

[52] V. Monjezi, A. Trivedi, V. Kreinovich, and S. Tizpaz-Niari, "Fairness testing through extreme value theory," in *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*, pp. 607–607, IEEE Computer Society, 2025.

[53] 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*, ISSTA 2021, (New York, NY, USA), p. 103–114, Association for Computing Machinery, 2021.

[54] S. Tizpaz-Niari, A. Kumar, G. Tan, and A. Trivedi, "Fairness-aware configuration of machine learning libraries," in *Proceedings of the 44th International Conference on Software Engineering*, pp. 909–920, 2022.

[55] R. K. Mothilal, A. Sharma, and C. Tan, "Explaining machine learning classifiers through diverse counterfactual explanations," in *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, FAT* '20, (New York, NY, USA), p. 607–617, Association for Computing Machinery, 2020.

[56] S. Wachter, B. Mittelstadt, and C. Russell, "Counterfactual explanations without opening the black box: Automated decisions and the gdpr," 2018.

[57] R. Guidotti, A. Monreale, S. Ruggieri, D. Pedreschi, F. Turini, and F. Giannotti, "Local rule-based explanations of black box decision systems," 2018.

[58] M. T. Ribeiro, S. Singh, and C. Guestrin, "Anchors: high-precision model-agnostic explanations," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18, AAAI Press, 2018.