Efficient Counterexample-Guided Fairness Verification and Repair of Neural Networks Using Satisfiability Modulo Convex Programming

Arya Fayyazi[1] , **Yifeng Xiao**[2] , **Pierluigi Nuzzo**[2] and **Massoud Pedram**[1]
[1]University of Southern California
[2]University of California, Berkeley
{afayyazi, pedram}@usc.edu, yifengx@berkeley.edu, nuzzo@eecs.berkeley.edu

Abstract

Ensuring fairness is essential for ethical decision-making in various domains. Informally, a neural network is considered fair if it treats similar individuals similarly in a given task. We introduce FaVeR (Fairness Verification and Repair), a framework for efficiently verifying and repairing pre-trained neural networks with respect to individual fairness properties. FaVeR ensures fairness via iterative search of high-sensitivity neurons and backward adjustment of their weights, guided by counterexamples generated from fairness verification using satisfiability modulo convex programming. By addressing fairness at the neuronal level, FaVeR aims to minimize the impact of neural network repair on overall performance. Empirical evaluations on common fairness datasets show that FaVeR achieves a 100% fairness repair rate across all models with an accuracy reduction of less than 2.27% and significantly lower average runtime than alternative repair methods.

1 Introduction

Deep neural networks (DNNs) are increasingly utilized to make critical decisions across various domains, including finance, healthcare, and law enforcement. These decisions significantly impact individuals' lives, making it imperative that the outcomes be fair and unbiased. However, models trained on historical data are often prone to biases based on protected attributes such as race, gender, and age. These biases can lead to unfair and discriminatory decisions, raising ethical and legal concerns. Addressing these concerns is essential not only for ensuring ethical artificial intelligence (AI) practices but also for complying with regulatory standards and maintaining public trust. Fairness properties in machine learning can be broadly classified into two categories: *group fairness* and *individual fairness*. Group fairness evaluates the behavior of the model for different demographic groups to ensure that no group is systematically disadvantaged [Albarghouthi *et al.*, 2017; Bastani *et al.*, 2019]. Metrics such as demographic parity, equal opportunity, and disparate impact fall into this category. However, enforcing group fairness can still lead to unfair treatment of individuals within the groups. In contrast, individual fairness ensures that individuals with similar unprotected attributes, such as qualifications or experience, receive similar outcomes, regardless of their protected attributes, e.g., age, race [Zhang *et al.*, 2020; Zheng *et al.*, 2022; Fayyazi *et al.*, 2025]. This concept is closely aligned with the principle of treating similar cases consistently and is essential for situations where decisions are made on an individual basis. This paper aims to ensure individual fairness of DNNs through a fairness verification and repair process.

Several methods have been proposed to verify the fairness of machine learning models. Techniques based on satisfiability modulo theories (SMT) [Benussi *et al.*, 2022] and mixed integer linear programming (MILP) [Biswas and Rajan, 2023; Mohammadi *et al.*, 2023] have been effective in verifying individual fairness properties. SMT-based techniques use logical constraints to ensure that the model behaves consistently across certain categories of inputs, while MILP-based approaches formulate the fairness verification problem as a MILP problem, where the presence of integer variables significantly increases computational complexity. As a result, these techniques may not scale well to large networks due to the combinatorial explosion of possible configurations and the high computational complexity involved.

However, researchers have employed various strategies to repair unfair models, including pre-processing, post-processing, and in-processing methods [Barocas *et al.*, 2023]. Pre-processing methods involve modifying the training data to remove biases before model training, while post-processing techniques adjust the model's predictions to ensure fairness after training. However, both strategies operate externally to the model's internal mechanics, often failing to correct biases learned in the latent representations, particularly in DNNs that learn complex, latent features. In-processing methods address this limitation by directly modifying the model's parameters during training or through *post-hoc* adjustments to incorporate fairness constraints. While effective, these methods must balance fairness improvements with computational efficiency and model accuracy.

In this context, we introduce a novel framework, named FaVeR (Fairness Verification and Repair), that efficiently addresses both the verification and repair of individual fairness properties in DNNs, as shown in Figure 1. Our approach is twofold. First, we encode the fairness verification prob-

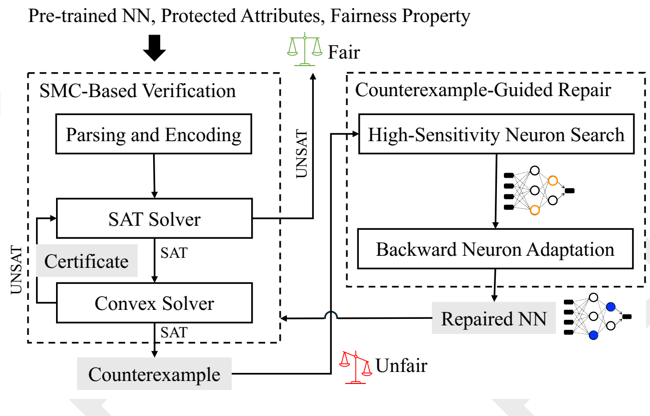

Pre-trained NN, Protected Attributes, Fairness Property
Fair
SMC-Based Verification
Parsing and Encoding
SAT Solver
UNSAT
Certificate
SAT
Convex Solver
SAT
UNSAT
Counterexample-Guided Repair
High-Sensitivity Neuron Search
Backward Neuron Adaptation
Repaired NN
Counterexample
Unfair

Figure 1: High-level view of the FaVeR workflow.

lem into a satisfiability modulo convex programming (SMC) problem, which efficiently integrates logical reasoning with numerical optimization in an iterative way. As a second step, if the network is unfair, FaVeR focuses on a targeted repair process guided by the counterexamples generated by the SMC procedure. High-sensitivity neurons are identified using a *sensitivity score*, which is calculated from a difference in activation values. A backward weight adaptation process then begins from the last DNN layers, iteratively modifying the weights of the high-sensitivity neurons until fairness is achieved or a maximum number of iterations is reached.

The main contributions of our work are summarized as follows: (i) We introduce FaVeR, a framework that efficiently verifies and repairs DNNs for individual fairness using an SMC-based encoding and a backward neuron adaptation method for high-sensitivity neurons. (ii) We characterize the fairness guarantees of our approach, iteratively incorporating new counterexamples and adjusting high-sensitivity neurons. (iii) We illustrate the applicability of FaVeR to a wide range of DNN architectures and datasets, showing that it is a versatile tool for fairness repair in various applications.

2 Preliminaries

We introduce the foundational concepts and formulate the problem addressed by FaVeR.

2.1 Neural Network

We define a neural network (NN) $f : \mathcal{X} \to \mathcal{Y}$ as a composition of linear functions and activation functions, where $\mathcal{X} \subseteq \mathbb{R}^M$ is a bounded input domain and $\mathcal{Y}$ is the output domain.

Definition 1 (Feedforward Function of a NN)**.** *Let ϕ denote the activation function, $l \in \{1, ..., L\}$, and let $\mathbf{W}^{(l,l-1)} \in \mathbb{R}^{n_l \times n_{l-1}}$ and $\mathbf{b}^{(l)} \in \mathbb{R}^{n_l}$ denote the weight matrix and bias vector for layer l. Then, the pre-activation and activation of the i-th neuron in layer l are, respectively:*

$$a_i^{(l)} = \sum_{j=1}^{n_{l-1}} W_{i,j}^{(l,l-1)}\, x_j^{(l-1)} + b_i^{(l)},\; l \in \{1, \dots, L\},$$

$$x_i^{(l)} = \phi(a_i^{(l)}),\; l \in \{1, \dots, L-1\}, \quad x_i^{(L)} = a_i^{(L)},$$

where the last layer is assumed to be linear, leading to the following expression for $l \in \{1, \dots, L-1\}$,

$$\mathbf{x}^{(l)} = \phi(\mathbf{a}^{(l)}) = \phi\big(\mathbf{W}^{(l,l-1)}\mathbf{x}^{(l-1)} + \mathbf{b}^{(l)}\big),$$

where ϕ is applied element-wise, $\mathbf{x}^{(0)} = \mathbf{x}_{in} = \mathbf{x}$, and $\mathbf{x}^{(L)} = \mathbf{a}^{(L)} = \hat{f}(\mathbf{x})$ denotes the logit output. In a classification problem, the output is given by $f(\mathbf{x}) = \arg\max_i\ \hat{f}_i(\mathbf{x})$. In a regression problem, we have $f(\mathbf{x}) = \hat{f}(\mathbf{x})$.

2.2 Individual Fairness

Intuitively, *individual fairness* requires that all "similar" inputs get "similar" outputs from the model. Formally, assume $\mathbf{x} = (x_1, x_2, \dots, x_M)^T$ is an instance from a dataset with M attribute values, where the set of attributes is denoted as $A = \{A_1, \dots, A_M\}$ and a^T denotes the transpose of $a \in \mathbb{R}^M$. The set of protected attributes (PA), such as `age` or `race`, is denoted by P, where $P \subset A$.

Definition 2 (Individual Fairness)**.** *The model f is individually fair if there exists no pair $(\mathbf{x}, \mathbf{x}')$ of data points in the input domain $\mathcal{X}$ such that: (1) $\forall \alpha, A_\alpha \in A \setminus P$: $x_\alpha = x'_\alpha$; (2) $\exists \beta, A_\beta \in P$: $x_\beta \neq x'_\beta$; and (3) $f(\mathbf{x}) \neq f(\mathbf{x}')$.*

This definition assumes a classification setting where the model output is discrete (e.g., a set of class labels) and the inputs associated with protected attributes are also discrete. In this context, differences are detected via value (label) mismatch. For regression or continuous outputs, condition (3) can be relaxed to a thresholded difference, e.g., $|f(\mathbf{x}) - f(\mathbf{x}')| > \delta$. Definition 2 [Dwork *et al.*, 2012] requires that two individuals with identical values for the unprotected attributes, but differing in the protected attributes, produce the same output. If this is not the case for a pair $(\mathbf{x}, \mathbf{x}')$, then we say that individual fairness is violated and $(\mathbf{x}, \mathbf{x}')$ is a counterexample for individual fairness. For example, suppose that we use NN to predict whether an individual's income exceeds `$50K`, with `race` designated as a protected attribute. According to individual fairness, if two individuals with identical attributes, such as `occupation` and `work experience`, but differing in `race`, are assigned different income predictions, the model is unfair. Definition 2 is often impractical in the real world due to its strict requirements on unprotected attributes. It can then be relaxed to the following notion of ϵ-fairness.

Definition 3 (ϵ-Fairness)**.** *The model f is individually ϵ-fair if there is no pair $(\mathbf{x}, \mathbf{x}')$ of data points in the input domain $\mathcal{X}$ such that: (1) $\forall \alpha, A_\alpha \in A \setminus P$: $|x_\alpha - x'_\alpha| \leq \epsilon_\alpha$; (2) $\exists \beta, A_\beta \in P$: $x_\beta \neq x'_\beta$; and (3) $f(\mathbf{x}) \neq f(\mathbf{x}')$.*

This relaxed definition [Biswas and Rajan, 2023] extends standard individual fairness by allowing small deviations in unprotected attributes, making it suitable for real-world applications where exact equality is unrealistic. For example, if two individuals who differ only slightly in an unprotected attribute but have different values in a protected attribute are assigned different predicted classes, the model is still considered unfair under ϵ-fairness.

2.3 Satisfiability Modulo Convex Programming

The fairness verification problem can be translated into a satisfiability problem for a class of first-order formulas over Boolean variables and convex constraints, which can be solved using the SMC approach. SMC was shown to be effective in handling combinatorial problems over discrete and continuous variables [Shoukry *et al.*, 2017; Shoukry *et al.*, 2018] by adopting a lazy coordination of a Boolean satisfiability (SAT) solver and a convex solver, inspired by the lazy SMT paradigm [Barrett and Tinelli, 2018]. In SMC, the SAT solver efficiently reasons about combinations of Boolean constraints to suggest possible assignments, while the convex solver checks their consistency and returns an infeasibility certificate when conflicts arise.

2.4 NN Fairness Verification and Repair

We formalize two core tasks: verifying whether a pre-trained NN satisfies a fairness property, and repairing the network if a violation is detected.

Problem 1 (NN Verification)**.** *Given a pre-trained NN model f and a set of protected attributes P, determine whether there exists a counterexample pair $(\mathbf{x}, \mathbf{x}') \in \mathcal{X}^2$ such that:*

- $|x_\alpha - x'_\alpha| \le \epsilon_\alpha$ *for all* $A_\alpha \in A \setminus P$,
- $x_\beta \ne x'_\beta$ *for at least one* $A_\beta \in P$,
- $f(\mathbf{x}) \ne f(\mathbf{x}')$.

Problem 2 (Verification-Guided Repair)**.** *Given a pre-trained NN model f and a counterexample pair $(\mathbf{x}, \mathbf{x}')$, find f' such that $f'(\mathbf{x}) = f'(\mathbf{x}')$, where f' is a NN obtained via fairness-oriented repair of f.*

By iteratively solving Problem 1 and Problem 2, FaVeR aims to construct a modified neural network that satisfies the ϵ-fairness property. We choose $\epsilon_\alpha = 0$ for individual fairness.

3 FaVeR: Fairness Verification and Repair

We detail our methodology, covering the SMC encoding of the fairness property and the verification problem as well as the repair procedure.

3.1 SMC-Based Verification

We encode the verification problem into a satisfiability problem for an SMC formula, and solve it by coordinating decision procedures based on SAT solving and convex programming. Based on Definition 1, for each neuron i in layer $l = 1, \dots, L-1$, we encode the ReLU activation $\phi(a_i^{(l)}) = \max(0, a_i^{(l)})$ using a Boolean variable $m_i^{(l)}$, which indicates the activation region:

$$\begin{aligned}\Big(m_i^{(l)} \rightarrow \big((a_i^{(l)} \ge 0) \wedge (x_i^{(l)} = a_i^{(l)})\big)\Big) \wedge \\ \Big(\neg m_i^{(l)} \rightarrow \big((a_i^{(l)} < 0) \wedge (x_i^{(l)} = 0)\big)\Big).\end{aligned} \tag{1}$$

This encoding is applied to all neurons across layers. The complete feedforward behavior is captured by:

$$\begin{aligned}\bigwedge_{l=1}^{L-1} \big(\mathbf{x}^{(l)} = \phi(\mathbf{a}^{(l)})\big) \;\wedge\; \big(\mathbf{x}^{(L)} = \mathbf{a}^{(L)}\big) \wedge \\ \bigwedge_{l=1}^{L} \big(\mathbf{a}^{(l)} = \mathbf{W}^{(l,l-1)}\mathbf{x}^{(l-1)} + \mathbf{b}^{(l)}\big).\end{aligned} \tag{2}$$

To verify fairness, we encode the constraints from Definitions 2 and 3 using Boolean variables. We introduce $m_i^{(0)}$ for each protected attribute A_β and $m^{(L)}$ for the output layer. We define:

$$\begin{aligned}\varphi_p &:= \bigwedge_{A_\beta \in P} \Big((m_\beta^{(0)} \rightarrow x_\beta = x'_\beta) \wedge (\neg m_\beta^{(0)} \rightarrow x_\beta \ne x'_\beta)\Big) \\ &\quad \wedge \big(\bigvee_{A_\beta \in P} m_\beta^{(0)}\big) \\ \varphi_u &:= \bigwedge_{A_\alpha \in A \setminus P} \Big((x_\alpha - x'_\alpha \le \epsilon_\alpha^{(l)}) \wedge (x_\alpha - x'_\alpha \ge -\epsilon_\alpha^{(l)})\Big), \\ \varphi_o &:= \Big(m^{(L)} \rightarrow f(\mathbf{x}) > f(\mathbf{x}')\Big) \wedge \Big(\neg m^{(L)} \rightarrow f(\mathbf{x}) < f(\mathbf{x}')\Big),\end{aligned}$$

where φ_p, φ_u, and φ_o enforce attribute constraints and output difference conditions for a fairness counterexample. To detect a fairness violation, we solve for an assignment to input pairs $(\mathbf{x}, \mathbf{x}')$ such that at least one of the protected attributes differs, the unprotected ones remain similar (within $\epsilon_i^{(l)}$), and the final output $f(\mathbf{x})$ differs from $f(\mathbf{x}')$. For individual fairness, we set all $\epsilon_\alpha^{(l)}$ to zero. We combine the constraints as

$$\Phi := \varphi_p \wedge \varphi_u \wedge \varphi_o, \tag{3}$$

and check the satisfiability of Φ in conjunction with the NN constraints in (1) and (2) using an SMC solver. The solver decomposes the problem into SAT and convex constraint feasibility subproblems over Boolean and real-valued variables, respectively. It then incrementally refines its search space by evaluating the feasibility of each Boolean assignment using convex programming to efficiently detect fairness counterexamples [Shoukry *et al.*, 2017; Naik and Nuzzo, 2020].

3.2 Counterexample-Guided Repair

When the SMC solver discovers a counterexample, FaVeR carries out a targeted repair process that refines the weights of the *high-sensitivity neurons* —neurons exhibiting significant difference in activation between the counterexample inputs $\mathbf{x}$ and $\mathbf{x}'$. Our procedure aims to iteratively adjust these weights starting from the last layer and moving backward until the network becomes fair.

High-Sensitivity Neuron Search

FaVeR identifies high-sensitivity neurons by comparing their activation values under inputs $\mathbf{x}$ and $\mathbf{x}'$. If the activation difference exceeds a threshold γ, then we obtain a candidate neuron for weight refinement.

Definition 4 (Sensitivity Score)**.** *Let $\mathcal{N} = \{n_i\}_{i=1}^N$ be the set of all neurons in the network, and let $\sigma_i(\mathbf{x})$ denote the* *activation of neuron n_i given input $\mathbf{x}$. For a counterexample pair $(\mathbf{x}, \mathbf{x}')$, the* sensitivity score *S_i of the i-th neuron is*

$$S_i \;:=\; \big|\sigma_i(\mathbf{x}) - \sigma_i(\mathbf{x}')\big|. \tag{4}$$

A neuron n_i is considered a high-sensitivity neuron if

$$S_i \;\geq\; \gamma, \quad \text{where} \quad \gamma \;=\; \tfrac{1}{2}\big(\max_{1\leq i\leq N} S_i + \min_{1\leq i\leq N} S_i\big).$$

This threshold highlights neurons with large activation deviation S_i relative to the overall range of sensitivity scores.

Backward Neuron Adaptation

Let $\mathcal{N}^{(l)}$ denote the set of neurons in layer l, $n_i^{(l)}$ the i-th neuron within the layer, and $S_i^{(l)}$ the corresponding sensitivity score. We apply the following *post-hoc* adaptation rule to each high-sensitivity neuron $n_i^{(l)}$, iterating over the layers from L down to 1:

$$\begin{aligned} W_{i,:}^{(l,l-1)} \;&\leftarrow\; W_{i,:}^{(l,l-1)} + \Delta W_{i,:}^{(l,l-1)}, \\ b_i^{(l)} \;&\leftarrow\; b_i^{(l)} + \Delta b_i^{(l)}, \\ \Delta W_{i,:}^{(l,l-1)} &= -\,\eta\,\lambda\,\mathrm{sign}\big(W_{i,:}^{(l,l-1)}\big)\, S_i^{(l)}\, W_{i,:}^{(l,l-1)}, \\ \Delta b_i^{(l)} &= -\eta\,\lambda\,\mathrm{sign}\big(b_i^{(l)}\big)\, S_i^{(l)}\, b_i^{(l)}, \end{aligned} \tag{5}$$

where $W_{i,:}^{(l,l-1)}$ denotes the row of weights associated with all the neurons in layer $l{-}1$ connecting to the i-th neuron in layer l, η is the step size, and λ controls the fairness-accuracy trade-off. These parameters are selected empirically on a held-out validation set to balance fairness improvement and accuracy retention, in the same spirit as cross-validation in training. We stop repairing once the accuracy drop exceeds a tolerance, so we usually apply the update only on a subset of K layers.

In the following, we show that the same amount of weight perturbation can produce a larger shift in the first-order logits of a neural network when performed to neurons that are closer to the output layer. Therefore, applying the adaptations in reverse layer order $\pi_r = (L, L{-}1, \ldots, 1)$ tends to maximize the impact on the norm of the first-order logit perturbation for any prefix length $K \leq L$, thus requiring fewer weight adjustments to achieve the desired unfairness reduction.

Theorem 1 (Logit Shift Maximization)**.** *Let $(\mathbf{x}, \mathbf{x}')$ be a counterexample and $\hat{f} : \mathcal{X} \to \mathbb{R}^{n_L}$ the network logit function. For each layer $l \in \{1, \ldots, L\}$ and neuron $i \in \{1, \ldots, n_l\}$, we define its* influence matrix slice

$$\boldsymbol{\mu}_i^{(l)} \;=\; \frac{\partial\big(\hat{f}(\mathbf{x}) - \hat{f}(\mathbf{x}')\big)}{\partial \mathbf{W}_{i,:}^{(l,l-1)}} \;\in\; \mathbb{R}^{n_L \times n_{l-1}},$$

capturing how perturbations to the incoming weights of neuron i affect the output logits [Goodfellow et al.*, 2016; Montavon* et al.*, 2018]. For a weight-slice update $\Delta W_{i,:}^{(l,l-1)} \in \mathbb{R}^{n_{l-1}}$, the corresponding first-order logit perturbation is*

$$\Delta\hat{\mathbf{y}}_i^{(l)} \;=\; \boldsymbol{\mu}_i^{(l)}\,\Delta W_{i,:}^{(l,l-1)}.$$

If neuron i in layer l and neuron j in layer $k < l$ receive updates of equal norm, i.e., $\|\Delta W_{i,:}^{(l,l-1)}\|_2 = \|\Delta W_{j,:}^{(k,k-1)}\|_2 = c$, then we obtain

$$\max_{\|\Delta W_{i,:}^{(l,l-1)}\|_2 = c} \big\|\Delta\hat{\mathbf{y}}_i^{(l)}\big\|_2 \geq \max_{\|\Delta W_{j,:}^{(k,k-1)}\|_2 = c} \big\|\Delta\hat{\mathbf{y}}_j^{(k)}\big\|_2.$$

Proof. By the *gradient attenuation* property of neural networks [Hochreiter, 1998; Bengio *et al.*, 1994], we obtain

$$\|\boldsymbol{\mu}^{(L)}\|_2 \geq \|\boldsymbol{\mu}^{(L-1)}\|_2 \geq \cdots \geq \|\boldsymbol{\mu}^{(1)}\|_2.$$

By the properties of spectral norms, we also have

$$\begin{aligned} \max_{\|\Delta W_{i,:}^{(l,l-1)}\|_2 = c} \big\|\Delta\hat{\mathbf{y}}_i^{(l)}\big\|_2 &= \max_{\|\Delta W_{i,:}^{(l,l-1)}\|_2 = c} \Big\|\boldsymbol{\mu}_i^{(l)}\,\Delta W_{i,:}^{(l,l-1)}\Big\|_2 \\ &= c\,\Big\|\boldsymbol{\mu}_i^{(l)}\Big\|_2 \end{aligned}$$

and similarly,

$$\max_{\|\Delta W_{j,:}^{(k,k-1)}\|_2 = c} \big\|\Delta\hat{\mathbf{y}}_j^{(k)}\big\|_2 = c\,\big\|\boldsymbol{\mu}_j^{(k)}\big\|_2.$$

Since $l > k$, we also have $\|\boldsymbol{\mu}_i^{(l)}\|_2 \geq \|\boldsymbol{\mu}_j^{(k)}\|_2$. Therefore, we conclude

$$\max_{\|\Delta W_{i,:}^{(l,l-1)}\|_2 = c} \big\|\Delta\hat{\mathbf{y}}_i^{(l)}\big\|_2 \geq \max_{\|\Delta W_{j,:}^{(k,k-1)}\|_2 = c} \big\|\Delta\hat{\mathbf{y}}_j^{(k)}\big\|_2.$$

□

Algorithm 1 details FaVeR. The algorithm begins with a pre-trained network f and produces a repaired network f'. Each iteration calls the SMC solver to find a counterexample $(\mathbf{x}, \mathbf{x}')$ (line 8). If none exists, fairness is certified and f' is returned (line 10). Otherwise, we compute the sensitivity score S_i for each neuron n_i, and select the high-sensitivity set of neurons $\mathcal{S}$ (lines 13–15). For each $n_i \in \mathcal{S}$, from layer L to 1, we check whether it has already been adapted (line 18). If not, its weight row $W_{i,:}^{(l,l-1)}$ and bias $b_i^{(l)}$ are updated using (5) (lines 20–22). If the resulting accuracy degradation stays within α, the update is accepted and n_i is added to the adapted set $\mathcal{A}$; otherwise, the change is reverted (lines 23–26). This process ensures fairness violations are resolved with minimal impact on the predictive performance and avoids redundant updates to previously repaired neurons.

3.3 Formal Guarantees of FaVeR

We discuss the formal guarantees offered by FaVeR. In each repair iteration, FaVeR updates the neurons with a significantly large influence on the NN output. The following theorem states that, for small weight perturbations, such updates strictly reduce model unfairness.

Theorem 2 (Local Unfairness Reduction)**.** *Let $\hat{f}_t : \mathcal{X} \to \mathbb{R}^{n_L}$ be the network's logit function after t repair steps, and let $(\mathbf{x}_t, \mathbf{x}'_t)$ be a counterexample for $\hat{f}_t$. We define the current logit difference and its unfairness as follows:*

$$\mathbf{d}_t = \hat{f}_t(\mathbf{x}_t) - \hat{f}_t(\mathbf{x}'_t) \in \mathbb{R}^{n_L}, \qquad U_t = \|\mathbf{d}_t\|_2.$$

Let $\mathcal{S}_t$ be the set of high-sensitivity neurons at iteration t. Suppose we pick one neuron $n_i \in \mathcal{S}_t$ in layer l and apply the update in (5)*. Let $\hat{f}_{t+1}$ and $\mathbf{d}_{t+1}$ be the logit function and difference after the update. Then, for small perturbations $\|\Delta W_{i,:}^{(l,l-1)}\|_2$, we obtain $U_{t+1} < U_t$, i.e., unfairness strictly decreases.*

Algorithm 1: Fairness Verification and Repair

```
**Definitions:**
$\mathcal{N}^{(l)}$: set of neurons in layer $l$, $l = 1, \ldots, L$; $\mathcal{N} = \bigcup_{l=1}^{L} \mathcal{N}^{(l)}$;
$\mathcal{C}$: set of counterexamples; $\mathcal{A}$: set of adapted neurons
**Input:** NN model $f$, protected attributes $P$, fairness property $\Phi$, parameters $\eta$, $\lambda$, max iterations $T$, accuracy tolerance $\alpha$
**Output:** Repaired classifier $f'$
$f' \leftarrow f, \mathcal{C} \leftarrow \emptyset, \mathcal{A} \leftarrow \emptyset$
**for** $t = 1$ **to** $T$ **do**
    $(\mathbf{x}, \mathbf{x}') \leftarrow \mathsf{SMC\_Solver}(f', \Phi, P)$
    **if** $\mathbf{x} = \texttt{None}$ **then**
        **return** $f'$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{(\mathbf{x}, \mathbf{x}')\}$
    **for** $(\mathbf{x}, \mathbf{x}') \in \mathcal{C}$ **do**
        $S_i \leftarrow |\sigma_i(\mathbf{x}) - \sigma_i(\mathbf{x}')|, \quad \forall n_i \in \mathcal{N}$
        $\gamma \leftarrow \frac{1}{2}(\max_i S_i + \min_i S_i)$
        $\mathcal{S} \leftarrow \{n_i \in \mathcal{N} \mid S_i \geq \gamma\}$
        **for** $l = L$ **downto** $1$ **do**
            **for** $n_i \in \mathcal{S} \cap \mathcal{N}^{(l)}$ **do**
                **if** $n_i \in \mathcal{A}$ **then**
                    **continue**
                $W_{\mathrm{old}} \leftarrow W_{i,:}^{(l,l-1)}, \;\; b_{\mathrm{old}} \leftarrow b_i^{(l)}$
                $W_{i,:}^{(l,l-1)} \leftarrow W_{i,:}^{(l,l-1)} + \Delta W_{i,:}^{(l,l-1)}$
                $b_i^{(l)} \leftarrow b_i^{(l)} + \Delta b_i^{(l)}$
                **if** $|\mathcal{L}_{\mathrm{acc}}(f') - \mathcal{L}_{\mathrm{acc}}(f)| < \alpha \mathcal{L}_{\mathrm{acc}}(f)$ **then**
                    $\mathcal{A} \leftarrow \mathcal{A} \cup \{n_i^{(l)}\}$
                **else**
                    $W_{i,:}^{(l,l-1)} \leftarrow W_{\mathrm{old}}, \;\; b_i^{(l)} \leftarrow b_{\mathrm{old}}$
**return** $f'$
```

Proof. Under the assumption of small weight perturbation, by a first-order Taylor expansion of $\hat{f}_t$ around the current weights, the change in logit difference becomes

$$\mathbf{d}_{t+1} - \mathbf{d}_t \approx \sum_{j=1}^{n_{l-1}} \underbrace{\left[\frac{\partial \hat{f}_t(\mathbf{x}_t)}{\partial W_{i,j}^{(l,l-1)}} - \frac{\partial \hat{f}_t(\mathbf{x}_t')}{\partial W_{i,j}^{(l,l-1)}} \right]}_{\mathbf{g}_{l,i,j} \in \mathbb{R}^{n_L}} \cdot \Delta W_{i,j}^{(l,l-1)}.$$

We now evaluate the change in the norm:

$$U_{t+1} - U_t \;=\; \|\mathbf{d}_{t+1}\|_2 \;-\; \|\mathbf{d}_t\|_2.$$

For small updates, the derivative of the unfairness norm along the $(\mathbf{d}_{t+1} - \mathbf{d}_t)$ direction gives

$$\|\mathbf{d}_{t+1}\|_2 - \|\mathbf{d}_t\|_2 \;\approx\; \frac{\mathbf{d}_t^\top}{\|\mathbf{d}_t\|_2} \left(\mathbf{d}_{t+1} - \mathbf{d}_t\right).$$

By substituting the Taylor approximation above, we obtain

$$U_{t+1} - U_t \;\approx\; \sum_{j=1}^{n_{l-1}} \underbrace{\frac{\mathbf{d}_t^\top}{\|\mathbf{d}_t\|_2} \, \mathbf{g}_{l,i,j}}_{\nu_{l,i,j} > 0} \; \Delta W_{i,j}^{(l,l-1)},$$

where $\nu_{l,i,j} > 0$ holds since a high-sensitivity neuron satisfies $\mathbf{d}_t^\top \, \mathbf{g}_{l,i,j} > 0$. Finally, plugging in the weight update formula from (5) yields

$$U_{t+1} - U_t \;\approx\; -\eta \, \lambda \, S_i \sum_{j=1}^{n_{l-1}} \nu_{l,i,j} \left| W_{i,j}^{(l,l-1)} \right| \;<\; 0,$$

since $\eta, \lambda, S_i, \nu_{l,i,j}, |W_{i,j}|$ are all positive, which leads to the (local) inequality $U_{t+1} < U_t$. □

Theorem 3 (Progress and Termination)**.** *Let $\hat{f}_t$ be the logit network at iteration t and let $\mathcal{N}$ be the set of neurons. Then, Algorithm 1 terminates after at most $|\mathcal{N}|$ accepted updates. At any iteration t, we have $\mathcal{L}_{\mathrm{acc}}(\hat{f}_t) \;\leq\; (1+\alpha)\, \mathcal{L}_{\mathrm{acc}}(\hat{f}_0)$. Moreover, if any (local) update is applied upon termination at some iteration t^*, we have $U_{t^*} < U_0$.*

4 Experimental Results

We discuss the empirical effectiveness of FaVeR.

4.1 Experimental Setup

FaVeR is built using Python, with models trained via the Keras APIs [Urban *et al.*, 2020]. In alignment with previous research [Biswas and Rajan, 2023], we utilized Z3 [De Moura and Bjørner, 2008] and Gurobi [Gurobi Optimization, LLC, 2023] to solve SAT and convex problems, respectively. All the experiments were carried out on a 3.4-GHz AMD EPYC 7763 64-core processor with 32 GB of memory.

Benchmarks. We evaluated FaVeR on three datasets, namely, Bank Marketing (BM), Adult Census (AC), and German Credit (GC), which were also used to evaluate Fairify [Biswas and Rajan, 2023], as well as on the Compas (CP) dataset from FairQuant [Kim *et al.*, 2024], as detailed in the column "Datasets" of Table 1. We relaxed individual fairness with $\epsilon = 0.01$. We used $\lambda = 0.01$ for the regularization parameter and $\alpha = 0.02$ for the accuracy tolerance to balance fairness and accuracy. These values were selected as they consistently achieved high repair rates while maintaining accuracy across the tested models.

Fairness Metrics. We evaluated fairness using two metrics adapted from prior work [Dwork *et al.*, 2012; Krishna *et al.*, 2022; Zhou *et al.*, 2003]:

- **Pairwise Disagreement Fraction (PD).** PD measures the fraction of randomly sampled similar input pairs $(\mathbf{x}_i, \mathbf{x}_j)$ whose predicted labels differ. A lower PD indicates fewer disagreements among similar samples, hence higher fairness.
- **Local Consistency (LC).** For each sample $\mathbf{x}_i$, LC measures the fraction of its k-nearest neighbors, computed in the input space of unprotected attributes, that share the same predicted label, and then averages this over all samples. Higher LC indicates more consistent local predictions.

We report both metrics before and after fairness repair, showing how effectively disagreements among similar inputs are reduced (PD) and local agreement is promoted (LC).

Datasets				Verification				Repair						
PA	**Model**	**#Layers**	**#Neurons**	**Fairify Ver.**	**Fairify Time (s)**	**FaVeR Ver.**	**FaVeR Time (s)**	**Init. Acc.**	**Final Acc.**	**Init. PD**	**Final PD**	**Init. LC**	**Final LC**	**Time (s)**
Age	BM-4	5	318	SAT	49.98	SAT	2.80	89.04%	86.88%	15.00%	0.33%	97.46%	99.97%	49.58
	BM-5	4	49	SAT	29.61	SAT	0.25	86.88%	86.88%	18.33%	0.33%	96.80%	99.97%	18.39
	BM-6	4	35	SAT	24.84	SAT	0.45	87.91%	86.88%	7.00%	0	99.18%	1	7.75
	BM-7	4	145	SAT	31.64	SAT	0.38	87.93%	87.78%	18.67%	0	94.24%	1	30.89
	BM-8	7	141	UNK	79.08	SAT	1.66	88.72%	86.88%	11.67%	0	98.64%	1	25.72
Sex	GC-1	3	64	SAT	17.08	SAT	0.50	71.33%	69.33%	28.33%	0	79.07%	1	5.70
	GC-2	3	114	SAT	47.33	SAT	3.68	71.33%	69.33%	34.00%	0	83.47%	1	140.36
	GC-3	3	23	SAT	14.18	SAT	0.08	71.37%	69.33%	51.33%	0	71.60%	1	1.04
	GC-4	4	24	SAT	17.64	SAT	0.11	70.67%	69.33%	6.67%	0	98.40%	1	0.23
	GC-5	7	138	UNK	51.53	UNSAT	26.11	69.33%	69.33%	0	0	1	1	26.11
Race	AC-1	4	45	SAT	32.26	SAT	0.65	81.24%	79.36%	0.03%	0	99.67%	1	8.96
	AC-2	3	121	SAT	126.45	SAT	10.12	84.48%	82.21%	7.33%	5.31%	97.71%	98.02%	16.50
	AC-3	3	23	SAT	57.56	SAT	1.07	84.25%	82.00%	7.00%	4.67%	97.76%	98.12%	8.43
	AC-4	4	221	UNK	128.45	UNSAT	11.18	76.09%	76.09%	1.67%	1.67%	99.94%	99.94%	11.18
	AC-5	4	149	SAT	118.08	SAT	19.57	73.37%	71.92%	3.77%	3.67%	99.27%	99.34%	40.58
	AC-6	4	45	SAT	37.89	SAT	1.45	82.77%	81.49%	0.37%	0	99.87%	1	9.37
	AC-7	7	145	SAT	58.46	SAT	3.32	81.03%	78.74%	6.00%	0	98.64%	1	22.52
	AC-8	4	10	SAT	34.26	SAT	0.49	82.15%	81.93%	0.01%	0	99.93%	1	5.98
	AC-9	4	40	SAT	35.36	SAT	1.72	83.95%	83.95%	13.33%	0	96.33%	1	7.74
	AC-10	6	20	SAT	70.67	SAT	5.91	83.80%	83.80%	1.00%	0	99.70%	1	11.19
	AC-11	6	40	UNK	55.40	SAT	3.51	84.04%	84.04%	1.67%	0.63%	98.89%	99.78%	12.94
	AC-12	11	45	UNK	114.19	SAT	11.61	77.70%	75.55%	47.67%	0	94.45%	1	25.66
Race	CP-1	2	24	SAT	27.11	SAT	0.37	72.03%	71.33%	7.03%	0	97.75%	1	0.58
	CP-2	5	124	SAT	63.24	SAT	1.02	72.68%	71.65%	1.33%	0	99.07%	1	1.70
	CP-3	3	600	UNK	1000+	UNSAT	1.42	72.14%	72.14%	0	0	1	1	1.88
	CP-4	4	900	UNK	1000+	SAT	1.23	73.65%	72.25%	3.31%	0	99.49%	1	1.94

Table 1: Benchmarks and experimental results for fairness verification and repair of NNs.

4.2 Individual Fairness Verification

Comparison with Fairify. The "Verification" columns in Table 1 present a comparison between our SMC-based verification method and Fairify [Biswas and Rajan, 2023] in terms of the verification results and runtime, showing the efficiency of SMC-based verification. Our method required, on average, approximately 20× less runtime than Fairify, while also delivering accurate verification results. In this context, SAT indicates that a counterexample exists, meaning that the model is unfair, and UNSAT denotes that the model is fair. UNK indicates cases where the solver could not conclusively determine fairness violations.

Scalability. For the CP dataset, the model sizes scale from 24 to 900. While Fairify fails to provide results for CP-3 and CP-4, FaVeR successfully solves these problems within a reasonable time frame, demonstrating its capability to handle more complex models.

4.3 Repair for Individual Fairness

The "Repair" column in Table 1 presents the fairness metrics before ("Init.") and after ("Final") repair, along with the total runtime of FaVeR applied to various models. FaVeR successfully repaired all models (the SMC solver output was UNSAT after the repair procedure) with acceptable execution times, while maintaining a low accuracy decrease.

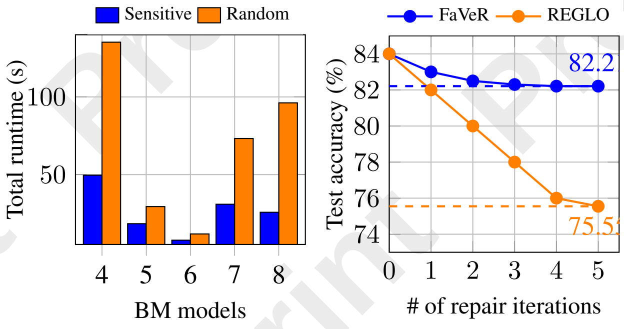
Effectiveness of the Neuron Selection Method. Figure 2a shows runtime comparisons on the Bank models, contrasting random neuron selection (orange bars) with our sensitivity-based neuron selection approach (blue bars) to repair. By targeting only the neurons that are most responsible for unfair outputs, our method typically reduces the time needed to reach a fair network, illustrating its practical gains.

Comparison with REGLO. Table 2 compares FaVeR with the state-of-the-art method REGLO [Fu *et al.*, 2024]. In our experiments, FaVeR consistently achieves a 100% repair rate with higher accuracy: 80.09% vs. 62.97% on AC, and 87.06% vs. 27.87% on BM. Both methods achieve 69.33% on GC. Figure 2b tracks the changes in accuracy of the repaired AC-2 model over repair iterations, showing that FaVeR can achieve a significantly smaller reduction in accuracy while attaining fairness in four repair iterations. It also achieves a shorter run-time in most models.

Fairness–Accuracy Trade-offs. FaVeR keeps accuracy loss low (average ≤2.27%, as shown in Table 1), but the exact loss can vary with model size and data. Networks with fewer neurons (e.g., AC-7) tend to lose more accuracy (2.29%) because small weight changes have a larger impact. Datasets where one class appears much more than others (e.g., GC-1) are harder to repair, since the model tends to favor the majority class; bigger changes are needed to correct unfair behaviors, which lead to greater accuracy loss. In models where features are highly interconnected (e.g., BM-4), even small changes can unintentionally affect multiple outputs. Using larger fairness parameters (λ, γ) incentivizes improving fairness, which can also reduce accuracy.

Model	Mean Initial Accuracy	FaVeR			REGLO		
		Mean Accuracy	Repair Rate	Mean Runtime	Mean Accuracy	Repair Rate	Mean Runtime
BM	88.14%	**87.06%**	**100%**	**26.47 s**	27.87%	60%	35.81 s
GC	71.60%	**69.33%**	**100%**	30.27 s	**69.33%**	**100%**	**1.75 s**
AC	82.33%	**80.09%**	**100%**	**15.09 s**	62.97%	**100%**	15.39 s

Table 2: Comparison of FaVeR and REGLO



(a)

(b)

Figure 2: (a) Runtime comparisons between random and sensitivity-based neuron selection approaches for FaVeR; (b) Comparison between FaVeR and REGLO on the AC-2 model: Changes in accuracy over repair iterations.

5 Related Work

This section reviews key literature on fairness verification and repair, which is related to FaVeR.

5.1 Individual Fairness vs. Group Fairness

Fairness properties in machine learning are often categorized as *individual fairness* and *group fairness*. Individual fairness requires that *similar individuals*, i.e., individuals differing only in protected attributes, receive *similar outcomes* [Dwork *et al.*, 2012; Bechavod and Ligett, 2017]. Group fairness, on the other hand, ensures equitable outcomes across protected groups using metrics like demographic parity and equalized odds [Hardt *et al.*, 2016; Hardt and Barocas, 2017]. While group fairness addresses population-level bias, it does not ensure fair treatment for every pair of similar individuals. Our work focuses on *individual fairness verification and repair*, ensuring consistent predictions for individuals with identical non-protected attributes.

5.2 Individual Fairness Verification

Our SMC-based method differs from existing techniques for verifying individual fairness properties of NNs, by virtue of its iterative approach efficiently coordinating SAT solving and convex programming to attack the complexity of the verification problem. SMT-based methods [Benussi *et al.*, 2022] verify individual fairness by formulating it as a satisfiability problem and leveraging SMT solvers. However, their scalability is limited for large networks. Similarly, MILP-based approaches [Biswas and Rajan, 2023; Mohammadi *et al.*, 2023] provide effective solutions but suffer from high computational costs, making them impractical for large-scale NNs.

5.3 Neural Network Repair

Methods for NN repair can generally be categorized into *pre-processing*, *in-processing*, and *post-processing* approaches. Our approach falls into the post-processing category, since we adjust the weights of specific, high-sensitivity neurons to enhance fairness. Pre-processing methods [Barocas *et al.*, 2023] mitigate bias by altering the training data distribution, for instance, through reweighting or feature transformation. In-processing methods [Dasu *et al.*, 2024; Li *et al.*, 2024; Gao *et al.*, 2022; Fu *et al.*, 2024] address bias by directly modifying the model while learning. For example, the adversarial training framework RUNNER [Li *et al.*, 2024] integrates fairness constraints during training using adversarial examples, although this increases computational overhead. Other post-processing techniques [Nguyen *et al.*, 2023; Li *et al.*, 2023] adjust weights in pre-trained models to improve fairness empirically, but they often rely on heuristic updates and lack a principled verification mechanism. Finally, REGLO [Fu *et al.*, 2024] proposes a provable repair method for ReLU networks by solving a convex program to compute minimal weight changes to satisfy fairness constraints. While effective, REGLO applies a global linearization of the network around specific inputs, which may overlook important nonlinearities of the model. Moreover, its objective function does not encourage preserving accuracy, e.g., via explicit regularization terms. This may lead to degraded performance, e.g., in our experiments, especially in deeper models where fairness conflicts with test accuracy and generalization.

6 Conclusion

We introduced FaVeR, a framework for the verification and repair of deep neural networks with respect to individual fairness properties. FaVeR leverages satisfiability modulo convex programming-based verification to guide the dynamic weight adaptation of high-sensitivity neurons, encouraging the satisfaction of the fairness specification with minimal impact on accuracy. Our theoretical analysis shows that the proposed adjustments effectively reduce the influence of high-sensitivity neurons on the overall fairness metric. Empirical results highlight the efficiency and effectiveness of FaVeR, achieving a 100% repair success rate in all the datasets tested, while maintaining high accuracy and low runtime.

Acknowledgments

This work was supported in part by NSF under awards 2514683 and 2514748.

References

[Albarghouthi *et al.*, 2017] Aws Albarghouthi, Loris D'Antoni, Samuel Drews, and Aditya V Nori. Fairsquare: probabilistic verification of program fairness. *Proc. ACM on Programming Languages (PACMPL)*, 1(OOPSLA):1–30, 2017.

[Barocas *et al.*, 2023] Solon Barocas, Moritz Hardt, and Arvind Narayanan. *Fairness and machine learning: Limitations and opportunities*. MIT press, 2023.

[Barrett and Tinelli, 2018] Clark Barrett and Cesare Tinelli. Satisfiability modulo theories. *Handbook of model checking*, pages 305–343, 2018.

[Bastani *et al.*, 2019] Osbert Bastani, Xin Zhang, and Armando Solar-Lezama. Probabilistic verification of fairness properties via concentration. *Proc. ACM on Programming Languages (PACMPL)*, 3(OOPSLA):1–27, 2019.

[Bechavod and Ligett, 2017] Yahav Bechavod and Katrina Ligett. Penalizing unfairness in binary classification. *arXiv preprint arXiv:1707.00044*, 2017.

[Bengio *et al.*, 1994] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE TNN*, 1994.

[Benussi *et al.*, 2022] Elias Benussi, Andrea Patane, Matthew Wicker, Luca Laurenti, and Marta Kwiatkowska. Individual fairness guarantees for neural networks. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, 2022.

[Biswas and Rajan, 2023] Sumon Biswas and Hridesh Rajan. Fairify: Fairness verification of neural networks. In *Proc. IEEE/ACM International Conference on Software Engineering (ICSE)*, pages 1546–1558, 2023.

[Dasu *et al.*, 2024] Vishnu Asutosh Dasu, Ashish Kumar, Saeid Tizpaz-Niari, and Gang Tan. NeuFair: Neural network fairness repair with dropout. In *Proc. ACM International Symposium on Software Testing and Analysis (ISSTA)*, pages 1541–1553, 2024.

[De Moura and Bjørner, 2008] Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 337–340. Springer, 2008.

[Dwork *et al.*, 2012] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proc. Innovations in Theoretical Computer Science Conference (ITCS)*, pages 214–226, 2012.

[Fayyazi *et al.*, 2025] Arya Fayyazi, Mehdi Kamal, and Massoud Pedram. FACTER: Fairness-aware conformal thresholding and prompt engineering for enabling fair llm-based recommender systems. *arXiv preprint arXiv:2502.02966*, 2025.

[Fu *et al.*, 2024] Feisi Fu, Zhilu Wang, Weichao Zhou, Yixuan Wang, Jiameng Fan, Chao Huang, Qi Zhu, Xin Chen, and Wenchao Li. REGLO: Provable neural network repair for global robustness properties. In *Proc. AAAI Conference on Artificial Intelligence*, volume 38, pages 12061–12071, 2024.

[Gao *et al.*, 2022] Xuanqi Gao, Juan Zhai, Shiqing Ma, Chao Shen, Yufei Chen, and Qian Wang. FairNeuron: Improving deep neural network fairness with adversary games on selective neurons. In *Proc. International Conference on Software Engineering (ICSE)*, pages 921–933, 2022.

[Goodfellow *et al.*, 2016] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. https://www.deeplearningbook.org/.

[Gurobi Optimization, LLC, 2023] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023.

[Hardt and Barocas, 2017] Moritz Hardt and Solon Barocas. Fairness in machine learning. In *Neural Information Processing Symposium, Tutorials Track*, 2017.

[Hardt *et al.*, 2016] Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. *Advances in neural information processing systems*, 29, 2016.

[Hochreiter, 1998] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets. *ICANN*, 1998.

[Kim *et al.*, 2024] Brian Hyeongseok Kim, Jingbo Wang, and Chao Wang. FairQuant: Certifying and quantifying fairness of deep neural networks. *arXiv preprint arXiv:2409.03220*, 2024.

[Krishna *et al.*, 2022] Satyapriya Krishna, Tessa Han, Alex Gu, Steven Wu, Shahin Jabbari, and Himabindu Lakkaraju. The disagreement problem in explainable machine learning: A practitioner's perspective. *arXiv preprint arXiv:2202.01602*, 2022.

[Li *et al.*, 2023] Tianlin Li, Xiaofei Xie, Jian Wang, Qing Guo, Aishan Liu, Lei Ma, and Yang Liu. Faire: Repairing fairness of neural networks via neuron condition synthesis. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 33(1):1–24, 2023.

[Li *et al.*, 2024] Tianlin Li, Yue Cao, Jian Zhang, Shiqian Zhao, Yihao Huang, Aishan Liu, Qing Guo, and Yang Liu. Runner: Responsible unfair neuron repair for enhancing deep neural network fairness. In *Proc. IEEE/ACM International Conference on Software Engineering (ICSE)*, pages 1–13, 2024.

[Mohammadi *et al.*, 2023] Kiarash Mohammadi, Aishwarya Sivaraman, and Golnoosh Farnadi. Feta: Fairness enforced verifying, training, and predicting algorithms for neural networks. In *Proc. ACM Conference on Equity and Access in Algorithms, Mechanisms, and Optimization (EAAMO)*, pages 1–11, 2023.

[Montavon *et al.*, 2018] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15, 2018.

[Naik and Nuzzo, 2020] Nikhil Naik and Pierluigi Nuzzo. Robustness contracts for scalable verification of neural network-enabled cyber-physical systems. In *Proc. IEEE/ACM International Conference on Formal Methods and Models for System Design (MEMOCODE)*, pages 1–12, 2020.

[Nguyen *et al.*, 2023] Giang Nguyen, Sumon Biswas, and Hridesh Rajan. Fix fairness, don't ruin accuracy: Performance aware fairness repair using automl. In *Proc. ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, pages 502–514, 2023.

[Shoukry *et al.*, 2017] Yasser Shoukry, Pierluigi Nuzzo, Alberto L Sangiovanni-Vincentelli, Sanjit A Seshia, George J Pappas, and Paulo Tabuada. SMC: Satisfiability modulo convex optimization. In *Proc. International Conference on Hybrid Systems: Computation and Control (HSCC)*, pages 19–28, 2017.

[Shoukry *et al.*, 2018] Yasser Shoukry, Pierluigi Nuzzo, Alberto L Sangiovanni-Vincentelli, Sanjit A Seshia, George J Pappas, and Paulo Tabuada. SMC: Satisfiability modulo convex programming. *Proc. IEEE*, 106(9):1655–1679, 2018.

[Urban *et al.*, 2020] Caterina Urban, Maria Christakis, Valentin Wüstholz, and Fuyuan Zhang. Perfectly parallel fairness certification of neural networks. *Proc. ACM on Programming Languages (PACMPL)*, 4(OOPSLA):1–30, 2020.

[Zhang *et al.*, 2020] Peixin Zhang, Jingyi Wang, Jun Sun, Guoliang Dong, Xinyu Wang, Xingen Wang, Jin Song Dong, and Ting Dai. White-box fairness testing through adversarial sampling. In *Proc. IEEE/ACM International Conference on Software Engineering (ICSE)*, pages 949–960, 2020.

[Zheng *et al.*, 2022] Haibin Zheng, Zhiqing Chen, Tianyu Du, Xuhong Zhang, Yao Cheng, Shouling Ji, Jingyi Wang, Yue Yu, and Jinyin Chen. Neuronfair: Interpretable white-box fairness testing through biased neuron identification. In *Proc. International Conference on Software Engineering (ICSE)*, pages 1519–1531, 2022.

[Zhou *et al.*, 2003] Dengyong Zhou, Olivier Bousquet, Thomas Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. *Advances in Neural Information Processing Systems (NeurIPS)*, 16, 2003.