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

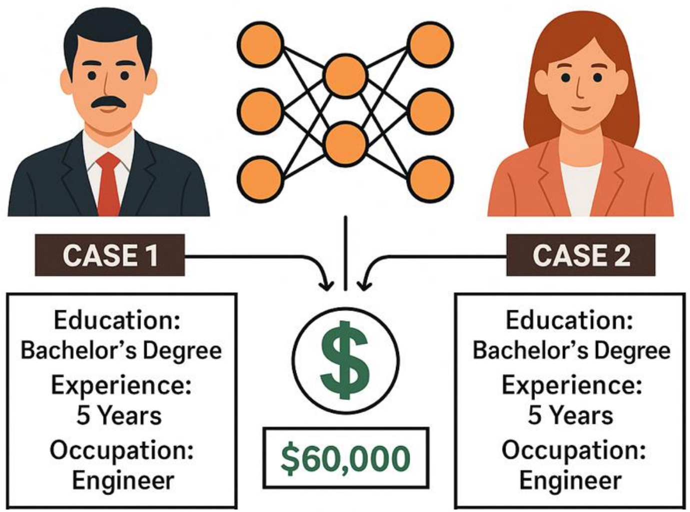
Arya Fayyazi[1] **Yifeng Xiao**[2] Pierluigi Nuzzo[2] Massoud Pedram[1]

[1]University of Southern California
[2]University of California, Berkeley

# *The Challenge: Ensuring Fair Decisions Made by Deep Neural Networks (DNNs)*

- DNNs increasingly drive high-stakes decisions for which fairness is essential.

- ***Individual Fairness***: Individuals with similar unprotected attributes receive similar outcomes, regardless of their protected attributes
  - Unprotected attributes: qualifications, experience
  - Protected attributes: age, race



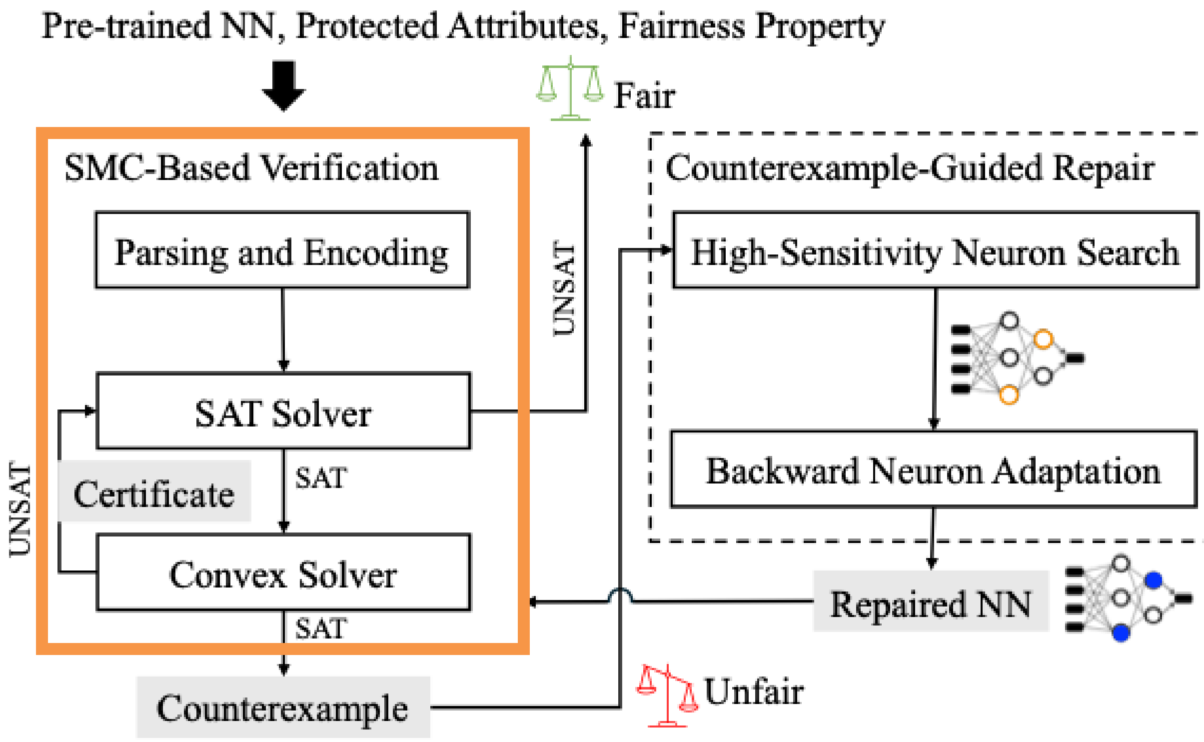

*We need effective methods for fairness verification and repair*

# Fairness of Neural Networks: Existing Approaches

- Verification
  - Satisfiability modulo theories (SMT)-based methods [Benussi et al., 2022]
  - Mixed integer linear programming (MILP) [Biswas and Rajan, 2023; Mohammadi et al., 2023]
- Repair
  - Pre-processing: Remove bias from training data [Barocas et al., 2023]
  - In-processing: Modify model parameters during training [Dasu et al., 2024; Li et al., 2024; Gao et al., 2022; Fu et al., 2024]
  - Post-processing: Adjust model predictions after training [Nguyen et al., 2023; Li et al., 2023; Fu et al., 2024]

*Our goal: More scalable verification and more efficient repair*

# FaVeR: Fairness Verification and Repair



# Individual Fairness

- Instance: $\mathbf{x} = (x_1, x_2, \dots, x_M)^T, \mathbf{x}' = (x_1{}', x_2{}', \dots, x_M{}')^T$
- Attributes: $A = \{A_1, \dots, A_M\}$; Protected attributes: $P \subset A$

*Individual Fairness*: No pair $(\mathbf{x}, \mathbf{x}')$ with

$$\forall \alpha \in A \backslash \mathrm{P} : x_a = x'_a, \qquad \exists \beta \in P : x_\beta \neq x'_\beta, \qquad f(\mathbf{x}) \neq f(\mathbf{x}')$$

Relax unprotected attributes

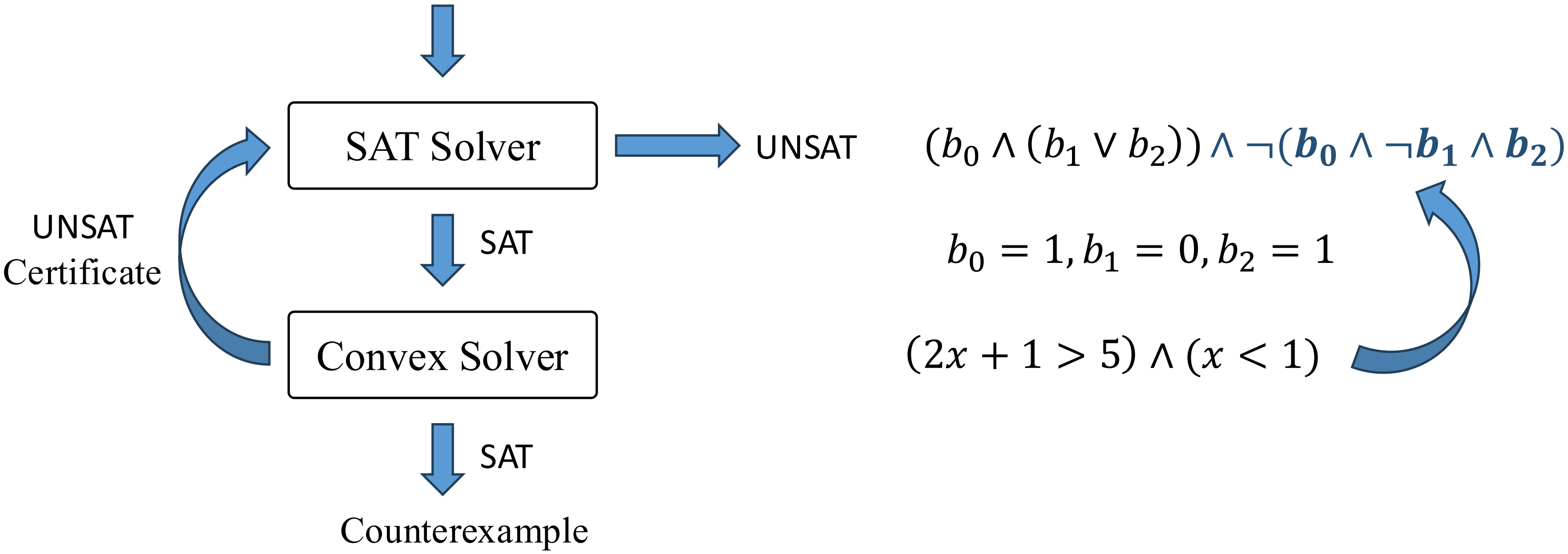
*$\epsilon$-Fairness*: $|x_\alpha - x'_\alpha| \leq \epsilon_\alpha$

*Verification: Check if $(\mathbf{x}, \mathbf{x}')$ exists with provided constraints*

# SMC-Based Verification

Check if $x$ exists for $(2x+1>5) \wedge ((x<4) \vee (x<1))$:

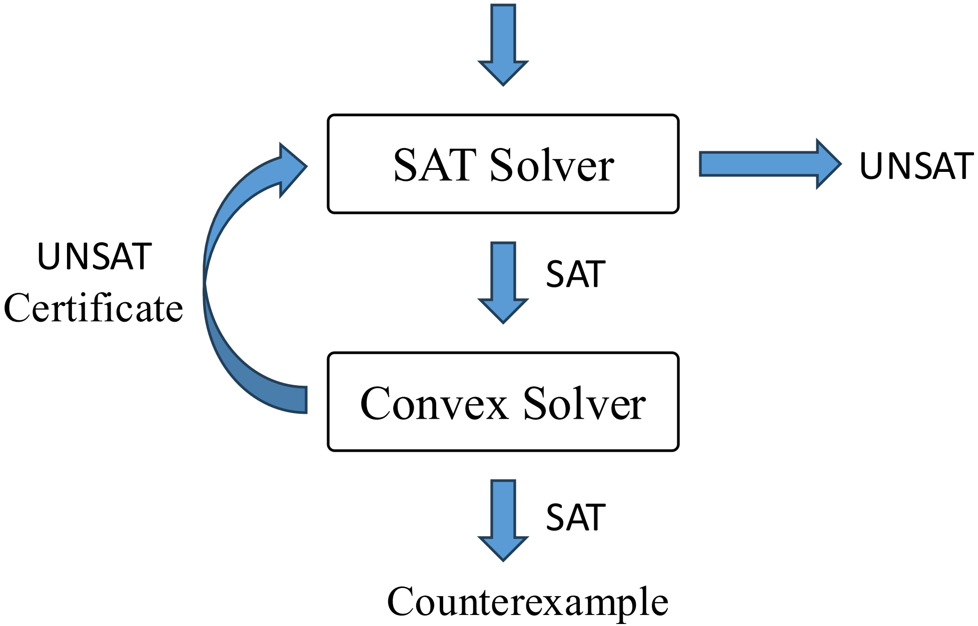
$$\left(b_0 \wedge (b_1 \vee b_2)\right) \wedge \left(b_0 \rightarrow (2x+1>5)\right) \wedge \left(b_1 \rightarrow (x<4)\right) \wedge \left(b_2 \rightarrow (x<1)\right)$$



# SMC-Based Verification

Check if $x$ exists for $(2x + 1 > 5) \wedge ((x < 4) \vee (x < 1))$:

$$\left(b_0 \wedge (b_1 \vee b_2)\right) \wedge \left(b_0 \rightarrow (2x + 1 > 5)\right) \wedge \left(b_1 \rightarrow (x < 4)\right) \wedge (b_2 \rightarrow (x < 1))$$



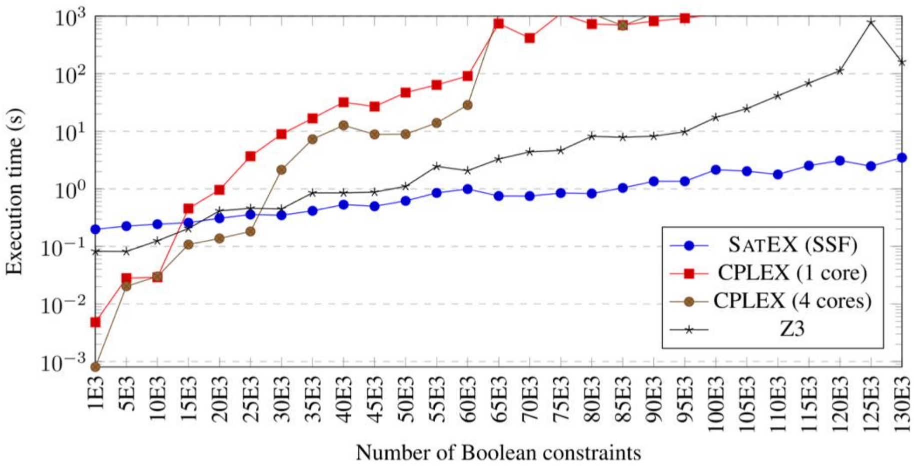

$$(b_0 \wedge (b_1 \vee b_2)) \wedge \neg(b_0 \wedge \neg b_1 \wedge b_2)$$

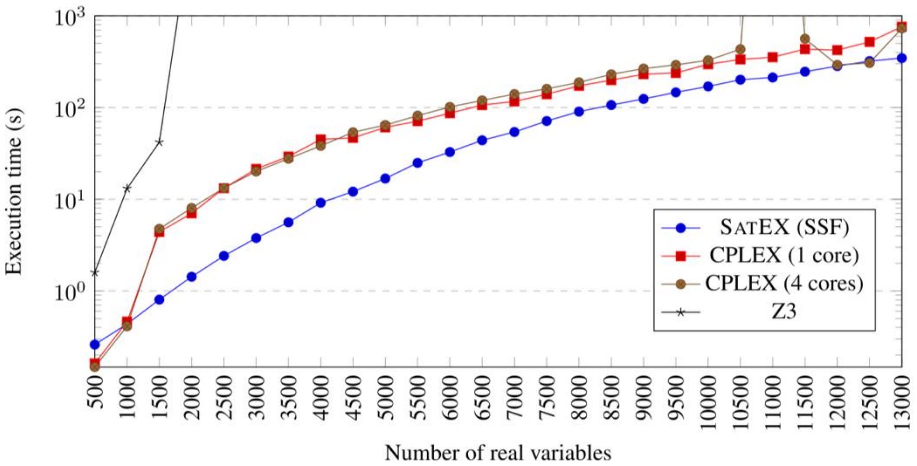
$$b_0 = 1, b_1 = 1, b_2 = 0$$

$$(2x + 1 > 5) \wedge (x < 4)$$

$$x = 3$$

# SMC-Based Verification




*SMC* [Shoukry et al., 2018] *is shown to outperform other methods for formulas with a large number of Boolean variables and convex constraints.*

# Problem Encoding

$\epsilon$-Fairness Property

$$\bigwedge_{A_\beta \in P} \Big( (m_\beta^{(0)} \rightarrow x_\beta = x'_\beta) \wedge (\neg m_\beta^{(0)} \rightarrow x_\beta \neq x'_\beta) \Big) \wedge ( \bigvee_{A_\beta \in P} m_\beta^{(0)} )$$

$$\bigwedge_{A_\alpha \in A \setminus P} \Big( (x_\alpha - x'_\alpha \leq \epsilon_\alpha^{(l)}) \wedge (x_\alpha - x'_\alpha \geq -\epsilon_\alpha^{(l)}) \Big)$$

$$\Big( m^{(L)} \rightarrow f(\mathbf{x}) > f(\mathbf{x}') \Big) \wedge \Big( \neg m^{(L)} \rightarrow f(\mathbf{x}) < f(\mathbf{x}') \Big)$$

Feedforward Behavior

$$\Big( m_i^{(l)} \rightarrow ((a_i^{(l)} \geq 0) \wedge (x_i^{(l)} = a_i^{(l)})) \Big) \wedge \Big( \neg m_i^{(l)} \rightarrow ((a_i^{(l)} < 0) \wedge (x_i^{(l)} = 0)) \Big)$$

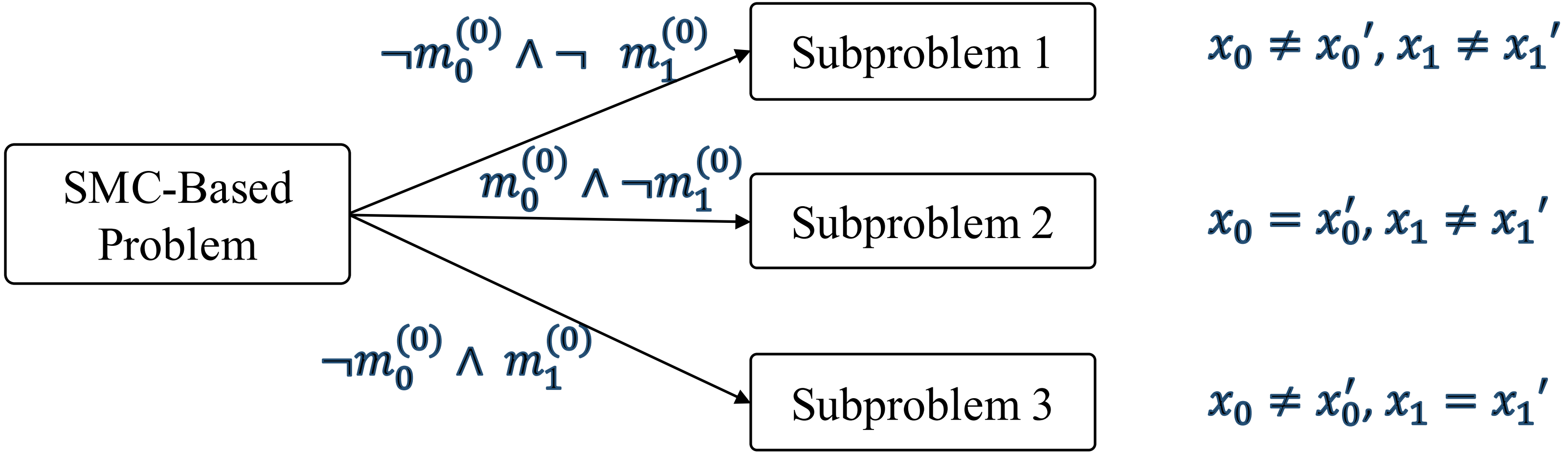
$$\bigwedge_{l=1}^{L-1} (\mathbf{x}^{(l)} = \phi(\mathbf{a}^{(l)})) \ \wedge \ (\mathbf{x}^{(L)} = \mathbf{a}^{(L)}) \wedge \bigwedge_{l=1}^{L} (\mathbf{a}^{(l)} = \mathbf{W}^{(l,l-1)} \mathbf{x}^{(l-1)} + \mathbf{b}^{(l)})$$

*Boolean variables $\boldsymbol{m}$ are introduced to encode conditional branching behavior.*

# Problem Decomposition

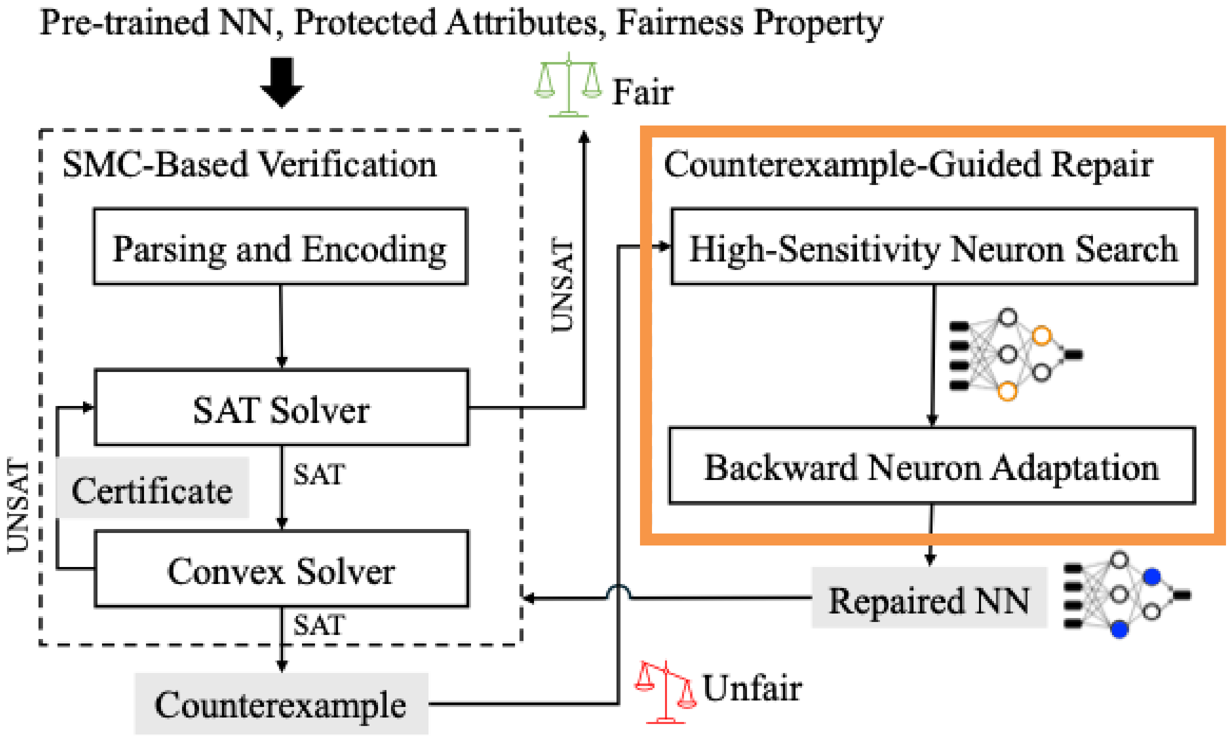
Introducing Boolean constraints enables decomposition of the verification problem.

Consider two protected attributes in the problem:




Add constraints in the SAT solver

# FaVeR: Fairness Verification and Repair



# High-Sensitivity Neuron Search

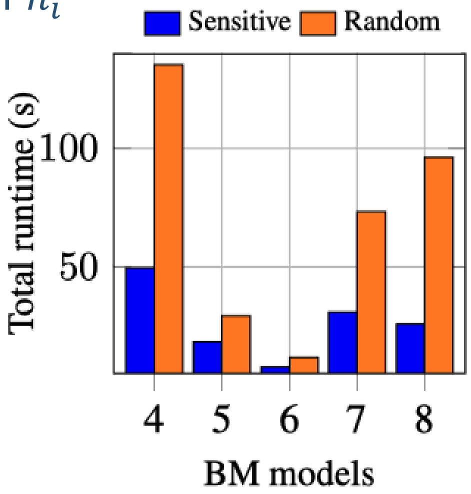
The activation of neuron $n_i$

- Sensitivity score for neuron:

$$S_i = |\sigma_i(\mathbf{x}) - \sigma_i(\mathbf{x}')|$$

Select high-sensitivity neurons with $S_i \geq \gamma$,

$$\gamma = \frac{1}{2}(\max_i S_i + \min_i S_i)$$




*Search for the neurons with high contributions to unfairness*

# Backward Neuron Adaptation

Unfairness: $U = ||f(\mathbf{x}) - f(\mathbf{x}')||$

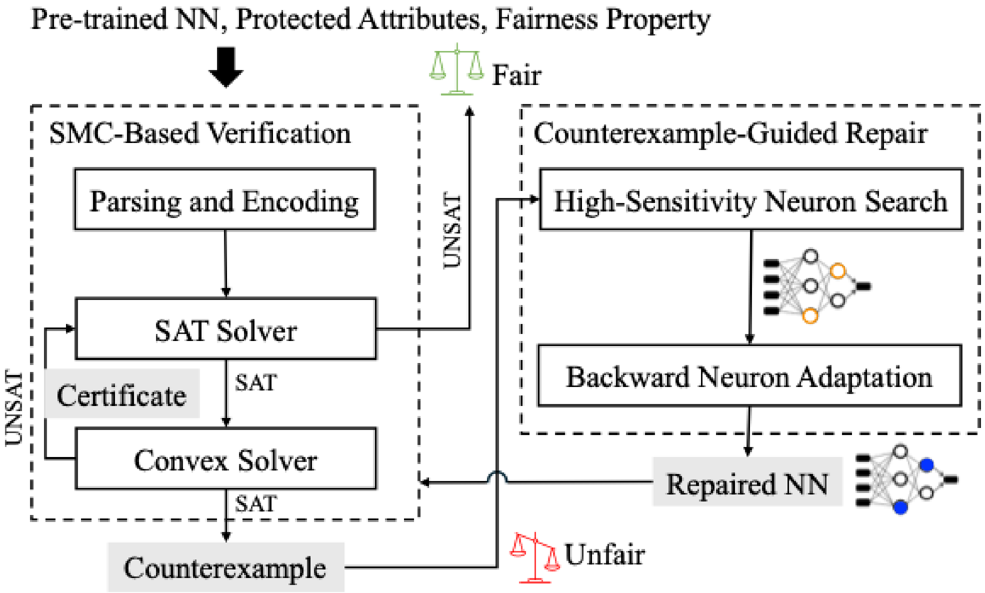
Update weights and bias of high-sensitivity neurons from layers $L$ to $1$:

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

*The same weight perturbation produces a larger shift in logits when applied to neurons closer to the output layer.*

*Each update reduces unfairness for small weights perturbations.*

# FaVeR: Fairness Verification and Repair




- Repair is rejected if accuracy drops below a specified threshold.
- The algorithm terminates when
  - The NN is fair, or
  - when all neurons have been adapted at most once.

# Experiments: Fairness Verification

Benchmark: Compas (CP) [Kim et al., 2024]

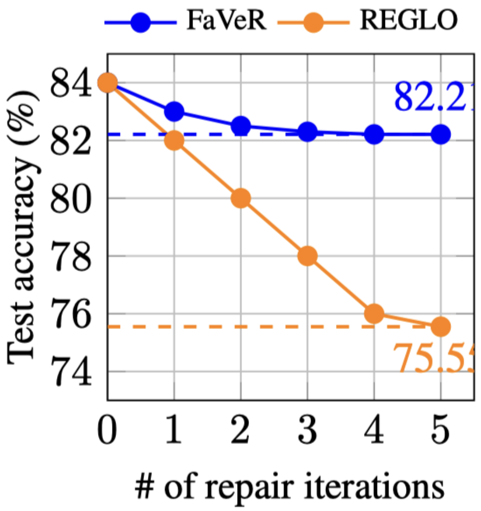
| *PA* | *Model* | *#Layers* | *#Neurons* | *Fairify Ver.* | *Fairify Time(s)* | *FaVeR Ver.* | *FaVeR Time (s)* |
|---|---|---|---|---|---|---|---|
| Race | CP-1 | 2 | 24 | SAT | 27.11 | SAT | **0.37** |
| | CP-2 | 5 | 124 | SAT | 63.24 | SAT | **1.02** |
| | CP-3 | 3 | 600 | **UNK** | 1000+ | UNSAT | **1.42** |
| | CP-4 | 4 | 900 | **UNK** | 1000+ | SAT | **1.23** |

*FaVeR is faster and solves cases unsolved by state-of-the-art comparable approaches (Fairify,* [Biswas and Rajan, 2023]*).*

# Experiments: Repair for Fairness

Comparison with REGLO [Fu et al., 2024]

Benchmarks: Bank Marketing (BM), Adult Census (AC), German Credit (GC)




| 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 |

*Efficient repair with less reduction in accuracy.*

# Conclusions