FETA: Fairness Enforced Verifying, Training, and Predicting Algorithms for Neural Networks

Kiarash Mohammadi  
Mila, Université de Montréal  
Canada

Aishwarya Sivaraman  
University of California, Los Angeles  
USA

Golnoosh Farnadi  
Mila, Université de Montréal  
HEC Montréal  
Canada

ABSTRACT
Algorithmic decision-making driven by neural networks has become very prominent in applications that directly affect people’s quality of life. This paper focuses on the problem of ensuring individual fairness in neural network models during verification, training, and prediction. A popular approach for enforcing fairness is to translate a fairness notion into constraints over the parameters of the model. However, such a translation does not always guarantee fair predictions of the trained neural network model. To address this challenge, we develop a counterexample-guided post-processing technique to provably enforce fairness constraints at prediction time. Contrary to prior work that enforces fairness only on points around test or train data, we are able to enforce and guarantee fairness on all points in the domain. Additionally, we propose a counterexample-guided loss as an in-processing technique to use fairness as an inductive bias by iteratively incorporating fairness counterexamples in the learning process. We have implemented these techniques in a tool called FETA. Empirical evaluation on real-world datasets indicates that FETA is not only able to guarantee fairness on-the-fly at prediction time but also is able to train accurate models exhibiting a much higher degree of individual fairness.

CCS CONCEPTS
• Human-centered computing → Collaborative and social computing systems and tools; Collaborative and social computing design and evaluation methods.

KEYWORDS
fairness, neural networks, verification

1 INTRODUCTION
Deep neural networks are increasingly used to make sensitive decisions, including financial decisions such as loan approval [24], recidivism risk assessments [30], salary prediction [4], etc. In these settings, for ethical, and legal reasons, it is of utmost importance that decisions are fair. For example, all else being equal, one would expect two individuals of a different gender to receive the same hiring decision. However, prior studies have shown that models trained on data are prone to bias on the basis of sensitive attributes such as race, gender, age, etc. [10, 42] It has been shown that even if sensitive features such as race and gender are withheld from the model, the model can still be unfair as it is often possible to reconstruct sensitive features that are encoded in data internally. Guaranteeing fairness not only helps organizations to address laws against discrimination but also helps users to better trust and understand the learned model [3].

Training neural network models such that fairness properties hold in their prediction is not always straightforward or possible. Existing approaches to the problem, either identify the absence of unfair predictions using verification [29, 62] or guarantee fairness only for points during the training phase by constrained optimization techniques [12] or add fairness regularizers to the loss function [33]. Other works which focus on test data points, provide fair models by using robustness techniques [54, 67]. While these techniques are successful in mitigating discrimination, they fail to provide global fairness guarantees for all points in the input domain at the prediction time. Over the past years, multiple definitions of fairness have been introduced. It is believed that none of these definitions dominates the others, and each of them is suitable for different settings. Recent works on fairness consider group-based notions of fairness [22, 69], e.g., demographic parity [17] or equalized odds [24], that indicate that two populations of individuals should be treated equally on average. Despite their prevalence, group fairness notions are generally hard to formally guarantee fairness for all input points [35, 54]. Further, an algorithm that satisfies group fairness could be blatantly unfair from the point of view of individual users [17]. In this paper, we focus on individual fairness [17] which states that the distance between the outcome for two individuals should be bounded according to the degree of their similarity.

Our approach. This paper develops techniques to detect, incorporate and guarantee individual fairness constraints for all points in the input space to a standard ReLU neural network without imposing further restrictions on the hypothesis space. These techniques leverage recent work that employs automated theorem provers to formally verify the properties of neural networks. We focus on the individual fairness notion introduced by [19]. This notion
says that a model is fair if, the decision of the model is the same for any two individuals with various combinations of sensitive attributes when nonsensitive attributes are fixed. To guarantee fairness, we present a counterexample-guided algorithm that detects and provably guarantees fairness at prediction time, given an arbitrary ReLU neural network. For any given model, our post-processing approach works by computing a majority decision for a group of individuals who share nonsensitive attributes on-the-fly via verification counterexamples. Furthermore, we propose a novel counterexample-guided algorithm to incorporate fairness during training. We identify individual fairness counterexamples on the training data, inducing additional supervision for training the network, and perform this process iteratively. We have implemented our algorithms in a tool called “Fairness Enforced Verifying, Training, and Predicting Algorithm” (FETA). Empirical evaluations on real-world benchmark datasets demonstrate the effectiveness of our solutions to train fair and accurate models, while provably guaranteeing fairness at the prediction time. Empirically, the two algorithms, when used in conjunction, enable better generalization while guaranteeing fairness.

Main contributions. Our key contributions are: 1) A practical individual fairness verification approach that detects discrimination through counterexamples given an arbitrary ReLU neural network (see Section 3: CE-Fair Verification). 2) A counterexample-guided online algorithm that provably guarantees individual fairness at prediction time (see Section 4: CE-Fair Prediction). 3) A counterexample-guided re-training algorithm that incorporates individual fairness during training (see Section 5: CE-Fair Training). 4) An end-to-end available implementation of our methods in an open-source tool called FETA, together with an extensive evaluation of real-world datasets (see Section 4.1 and Section 5.1).

2 PRELIMINARIES

We begin by introducing some common notations. Let $X$ be the input space consisting of $d$ features where $X \equiv N \times S$ and $S$ denotes protected or sensitive features, and $N$ denotes remaining input features, and suppose that it is a compact finite subset $X \equiv [L, U]^d$ of $N \in \mathbb{R}^k$ and $S \in \mathbb{N}^{d-k}$. Let $Y \in \{0, 1\}$ be the output space. We consider the supervised binary classification task, where $f_\theta : X \rightarrow \{0, 1\}$ outputs a probability distribution over classes, $\theta$ denotes the parameters and the classifier $g_\theta : f_\theta(X) \rightarrow Y$ is defined as $g_\theta = 1(f_\theta(x) \geq \Delta)$ where $\Delta$ is some classification threshold and it assigns an input to a category identified by numeric code $Y$. Let $D = \{(x_1, y_1), \ldots, (x_M, y_M)\}$ be the training dataset containing $M$ samples with $x_m$ and $y_m$ respectively denoting the $m$th individual and the corresponding output. The most commonly used Binary Cross Entropy (BCE) loss for this task is:

$$L_{BCE}(f_\theta) = -\frac{1}{M} \sum_{i=1}^{M} y_i \log(f_\theta(x_i)) + (1 - y_i) \log(1 - f_\theta(x_i))$$

where the goal is to find the best $\theta$ that minimizes $L_{BCE}$ across the data-generation distribution rather than just over the finite $D$.

Our goal will be to verify, guarantee and train an individually fair model in some sensitive input features. We refer to the Causal Discrimination definition, a notion of individual fairness proposed by [19].

Definition 2.1. (Causal Discrimination) Assume a function $g_\theta : f_\theta(X) \rightarrow Y$, such that $X[1 \ldots k] \in N$ and $X[k + 1 \ldots d] \in S$. We define $g_\theta$ to be individually fair in sensitive features $S$ iff for any two points $x, x' \in X$ where $x[i] = x'[i], \forall i \in \{1 \ldots k\}$, we have that $g_\theta(x) = g_\theta(x')$.

In Neural Networks (NN), various nonlinear activation functions have been introduced. Among those, ReLU has been used widely and generalized well [15, 21, 66], particularly in the context of verification [28, 34] and robustness. Hence, we will assume $f_\theta$ is a ReLU neural network. Formal properties of neural networks are often verified by encoding the semantics of neural networks ($f_\theta$) as logical constraints. While several approaches to encoding neural networks for verification have been studied [9, 43]; we use the encoding and optimization approach similar to [47] that propose significantly faster techniques for our use-case than the ones relying on Satisfiability Modulo Theories. This is crucial to our goal of guaranteeing fairness at prediction time.

Definition 2.2. (MLP Encoding of Neural Networks) Let $f_\theta$ be an $n$-layer fully-connected ReLU neural network with a single output where $\theta = (W, b)$ and $W_i$ and $b_i$ respectively denote weights and bias of $i$th layer. The width of each layer is represented by $t_i$, the values of neurons before applying ReLU are represented by vector $z_i, \forall i \in \{0 \ldots n\}$ ($z_0$ being the input), and their values after ReLU by $\hat{z}_i, \forall i \in \{1 \ldots n\}$, [60] proposes the following MLP encoding, $\forall i \in \{1 \ldots n\}$:

$$z_i = W_i \hat{z}_{i-1} + b_i$$

$$\delta_i \in \{0, 1\}^t_i, \quad \hat{z}_i = \max(0, z_i - \delta_i)$$

Equation (2a) encodes the linear relationship, while Equation (2b) encodes the ReLU activation function, i.e., $\hat{z} = ReLU (z) = \max(0, z)$. $\delta_i$ is a vector of binary variables representing the state of each ReLU as non-active or active. This encoding relies on bounds on the values of neurons, $I_i, U_i$. These bounds are computed using a linear approximation of the network proposed by [18], given the bounds on input $I_0 = L, U_0 = U$. Moreover, in this work, we assume both continuous and discrete domains over the input variables, hence the mixed-integer linear program (MILP).

Formal properties of functions are often characterized in terms of their counterexamples, e.g., [13, 58, 59]. The techniques proposed in this paper will be centred around using counterexamples to the fairness specification. Counterexample-guided algorithms rely on the ability to find counterexamples, which require that both the counterexample specification and the object of interest ($f_\theta$) to be encoded in a language amenable to automated reasoning. Definition 2.2 provides such an encoding for $f_\theta$. In the next section, we show how to add fairness constraints per Definition 2.1, to identify counterexamples.

3 CE-FAIR VERIFICATION

In 2019, Apple launched a credit card application that was accused of gender bias [5]. The scandal went viral when a couple who shared all of their bank accounts, assets, and credit cards, received different credit limits while only their gender was different in their applications. Inspired by this real-world example, in this section,
we propose an approach that focuses on auditing and verifying a trained neural network model to detect such discriminatory outcomes. We envision scenarios where the classifier is a proprietary model, belonging e.g. to a company or a bank, and an external party wants to inspect the model to ensure that it is operating fairly. We introduce Counterexample-guided fair (CE-FAIR) verification that can identify these fairness violations.

**Definition 3.1.** (CE-FAIR Verification) Consider example $x \in X$, function $f_0 : X \rightarrow [0, 1]$ and sensitive features $S$ and non-sensitive features $N$. Then a fairness counterexample for example $x$, function $f_0$, and $S$ is $x'$ such that (i) $x[i] = x'[i]$, $\forall i \in \{1 \ldots k\} \in N$, and (ii) $g_0(x) \neq g_0(x')$. We then define the CE-FAIR verification function $v$ for sensitive feature set $S$ that takes as input a function $f_0 : X \rightarrow [0, 1]$ and $x \in X$ as follows:

$$v(f_0, x) = \begin{cases} x' & \text{where } x' \text{ is a fairness counterexample} \\ \emptyset & \text{if no fairness counterexample exists} \end{cases}$$

CE-FAIR verification function $v(f_0, x)$ can find counterexample $x'$ by solving the optimization problem in which the two kinds of constraints defined in Definition 3.1 are added to the MILP encoding from Equation 2. The feasible set of this optimization problem is explored using an optimizer backend and if there exists a solution satisfying these constraints, we will have a fairness counterexample. Formally, the following constraints will be added to the MILP formulated in Equations 2a, 2b to encode fairness counterexamples:

$$z_0[i] = x[i], \forall i \in \{1 \ldots k\} \tag{3a}$$

$$1(z_n \geq \Delta) = 1 - g_0(x) \tag{3b}$$

where $z_0[i]$ is the variable associated with the $i$-th neuron in layer 0 (input layer). Concretely, this fairness verification approach searches for a counterexample with the same nonsensitive features as $x$ and any assignments to sensitive features ($z_{0,i}$ where $i \in \{k \ldots d\}$), constraining the output of the model to be opposite to $g_0(x)$. Here we only search for a counterexample violating fairness as a constraint by adding the following as an objective to the MILP:

$$|z_n - f_0(x)| \tag{4}$$

However, we can also search for the counterexample with maximum violation (see Definition 5.1). While CE-FAIR verification is sufficient to audit and verify a trained model to identify counterexamples as per Definition 2.1, in the case where there are counterexamples—which is often the case—it is not clear how to guarantee fairness, or how to enforce it during training. The next two sections present the counterexample-guided algorithms that address these challenges.

## 4 CE-FAIR PREDICTION

For ethical or legal reasons, a company may want to ensure that their model outcome is the same for all individuals irrespective of their sensitive attributes. Further, they may want to ensure fair predictions using techniques that do not require modifying and re-training the model under study. Such requirements can be due to the utilization of outsourced models without having any access to the original training data or having limited resources to re-train the model. In this section, we leverage our counterexample-guided verification approach to provably guarantee individual fairness at prediction time without any requirements to re-train the model. We propose an online technique that leverages counterexamples to Definition 2.1 to construct fair predictions on-the-fly at prediction time.

A naive approach to ensure fair predictions would be to return the same output for all individuals, e.g., the most frequent label in the training set. While this satisfies individual fairness, it leads to poor model performance (see Table 1 in Section 4.1). However, this gives us intuition to return the majority decision for a group of individuals who share nonsensitive attributes. We define CE-FAIR prediction that produces individually fair output for a given input $x$ and $f_0$ as:

**Definition 4.1.** (CE-FAIR Prediction) For an example $x \in X$, function $f_0 : X \rightarrow [0, 1]$, we define a post-processing CE-FAIR Prediction function $h$ such that:

$$h(f_0(x)) = \begin{cases} \frac{1}{|A(x)|} \left( \sum_{x' \in A(x)} g_0(x') - \sum_{g_0(x') = 0} g_0(x') \right) & \geq 0 \end{cases} \tag{5}$$

where:

$$A(x) = \{x | x[1] = x[1], \ldots, x[k] = x[k]\},$$

$$X[k+1] = a_k+1, \ldots, X[d] = a_d,$$

$$\forall a_k+1, \ldots, a_d \in [L'_k+1, \ldots, U'_k+1, \ldots, d]^{d-k}$$

Basically, this is contrasting the 0 and 1 outputs of $g_0$ in the space of sensitive features; the counter increments when $g_0(x')$ is 1 and decrements otherwise.

**Theorem 4.2.** For any function $f_0$ and for any input $x \in X$ with $S$ as sensitive features, $h(f_0(x))$ is individually fair in $S$.

**Proof.** The proof is trivial: $h$ outputs the same decision for all points within the group of all assignments to the sensitive attributes given fixed nonsensitive attributes of $x$, thus, no fairness counterexample exists. \hfill $\square$

So far we have established a way to guarantee fair predictions for all input points based on the majority decision captured in function $h$. To identify the majority decision, the simple approach is to enumerate all possible assignments of sensitive attributes. Concretely, given a test point $x$, we could traverse all possible assignments to the sensitive features, counting the frequency of each label. The computational complexity of this approach grows with the size of sensitive attributes and the domain size of each sensitive attribute. This approach, however, is not practical and increases prediction time significantly (See Appendix). This motivates the next approach in which we compute $h$ and identify the majority decision by leveraging our MILP framework to find counterexamples.

**Definition 4.3.** (CE-FAIR Counting) All counterexamples of a test sample $x$ (i.e., $S$) can be determined in an iterative way by adding the following constraints to the verification problem in
Algorithm 1 Counterexample-guided Counting to Guarantee Fair Predictions

**Input:** \( f_0, x \)

**Output:** CE-Fair Prediction: \( h(f_0(x)) \in \{0, 1\} \)

\[
\phi_N \leftarrow \text{ModelMIPEncoding}(f_0) \quad \text{[Constraints in Equation 2]}
\]

\[
\phi_{CE} \leftarrow \text{FairCEEncoding}(\phi_N, x, f_0(x)) \quad \text{[Constraints in Equation 3]}
\]

\[
S \leftarrow \text{FairCECounting}(\phi_N, \phi_{CE}) \quad \text{[Constraints in Equation 7]}
\]

if \(|S| < \left\lceil \frac{|A(x)|}{2} \right\rceil\) then

\[
\text{return } g_0(x)
\]

else

\[
\text{return } 1 - g_0(x)
\]

end if

Definition 3.1 and solve in iteration \( K + 1 \) as:

\[
x^i = u(f_0, x) \quad (7a)
\]

\[
\sum_{i=1}^{d} \left( \sum_{s \in S \ x^i_s \neq 0} x^i_s \right) \quad (7b)
\]

\[
k = 1, \ldots, K
\]

where \( S \) is a set of all counterexamples of \( x \) and \( x^k \) is a counterexample of \( x \) which is included in the set \( S \) from the previous iteration \( k \). To satisfy Constraints 7c, the solution must differ in at least one entry for each \( x^k \). Once Problem 7 becomes infeasible, then all counterexamples have been determined. Since there is a finite number of feasible assignments (e.g., Equation 6), this iterative method will stop in a finite time.

The lazy constraints generation approach defined in Definition 4.3 can be implemented more efficiently by using the optimization backend [23]. Hence, instead of iteratively finding counterexamples, we explore the MILP search tree in pursuit of \( \left\lceil \frac{|A(x)|}{2} \right\rceil \) counterexamples (rather than only one) where their labels are opposite to \( g_0(x) \). If that many solutions are found, then the majority decision for the group of assignments specified by \( x \) is opposite to \( g_0(x) \), otherwise, the prediction remains unchanged. The general scheme of CE-Fair Counting approach is shown in Algorithm 1. The algorithm takes as an input \( f_0 \), as well as a sample \( x \). In line 3, the MILP encoding of \( f_0 \) is obtained as per Equation 2 and constraints from Equation 3 specifying a counterexample for \( x \) are obtained in the following line. In line 5, the MILP search tree is explored to find counterexamples; if it finds less than \( \left\lceil \frac{|A(x)|}{2} \right\rceil \), the final prediction does not change, otherwise, it flips (lines 6–9).

4.1 Empirical Evaluation of CE-Fair Prediction

This section shows the effectiveness of CE-FAIR prediction approach through empirical evaluations on three widely known real-world benchmark datasets: German [27], IPUMS Adult, and Law School [63].

4.1.1 Dataset details. Here we overview the datasets we leverage:

- **German credit dataset [27]**
  This dataset consists of 1k samples with dimensionality 61 and sensitive features: age \( \in \{19, 75\} \), sex/marital status with 4 categories, and foreign worker with 2 categories. The main task is binary classification of good or bad credit risks.

- **IPUMS Adult dataset (aka, the new Adult) [14]**
  The initial Adult dataset [37] is used for binary classification of whether an individual’s salary is above or below $50k. [14] discuss some limitations of this dataset and propose a reconstruction of the Adult [37] dataset in which the actual income of the individuals are available. Thus, one can redefine the binary classification task with some threshold other than $50k. In our experiments, this threshold is set to $30k as the experiments by [14] indicate the most severe unfairness to occur around the 30k threshold.

  The dataset consists of 49k samples with dimensionality 103 and sensitive features: age \( \in \{17, 90\} \), marital status with 7 categories, race with 5 categories, native country with 41 categories, and sex with 2 categories. This is the largest sensitive feature space among datasets used for our experiments.

- **Law School dataset [63]**
  This dataset, consisting of 86k samples, gathers law school admission records and is used for predicting if an individual would pass the bar exam. The input dimension is 37 and the sensitive features are: race with 3 categories and gender with 2 categories

4.1.2 Experimental Setup. Experiments are implemented in Python using Pytorch [49]. All experiments were run on a machine with 10 GiB RAM and a 2.1GHz Intel Xeon processor. We use Gurobi-9.5.1\(^1\) as our backend solver to generate counterexamples. We make all code, datasets, and preprocessing pipelines publicly available. Below, we overview the experimental setup.

- **Data**
  The data is divided into 5 folds of 80/20 train/test sets. Moreover, 10% of the train set is sliced for validation. Experiment results are gathered with 5-fold cross-validation (CV). As for data types, we support categorical and numerical features. For the CE-Fair Prediction part, where we have counting over individuals sharing sensitive attributes, numerical features are considered discrete. Numerical features of nonsensitive attributes are considered real-valued.

- **Model Architecture**
  The ReLU neural network model used across all experiments is a fixed architecture of 3 hidden layers of width 16. This is a reasonably complex model for the tabular datasets used in such scenarios.

- **Pre-training**
  To pre-train the initial model (\( \text{NN}_h \)), we run a grid search over learning rate \((10^{-2}, 10^{-3}, 10^{-4})\) and batch size \((64, 128)\). We train each configuration for 500 epochs and select the model with the best loss on the validation set. This is the case with all datasets except for Law School which is more tricky to train on; for that, we use a learning rate of 0.01, a batch size of 256, train for 100 epochs, and take the last epoch model to get the best initial pre-trained model for a fair comparison.

\(^{1}\)https://www.gurobi.com
Q1: Does a deep neural network trained on data obey individual fairness? To quantify the degree of unfairness in the initial model trained on data, we introduce, Counterexample Rate (CE Rate), which computes how many test samples have counterexamples as per Definition 3.1. As shown in Figure 1, the degree of fairness violation based on these metrics is high for all our datasets, motivating the need for guaranteed fair predictions. The percentage of data points that have counterexample can be as high as 89% for IPUMS Adult dataset.

Table 1: The effect of guaranteed fair predictions on performance and model fairness – 5-fold CV results

| Dataset  | NNₚ | Majority Baseline | CE-Fair Prediction |
|----------|-----|------------------|--------------------|
|          | Accuracy | Flip Rate | Accuracy | Flip Rate | Accuracy | Flip Rate |
| German   | 76.70 ± 2.78 | 9.00 ± 0.73 | 7.00 ± 0.00 | 0.00 ± 0.00 | 74.20 ± 3.50 | 0.00 ± 0.00 |
| IPUMS Adult | 81.57 ± 0.47 | 24.39 ± 1.52 | 54.61 ± 6.55 | 0.00 ± 0.00 | 73.25 ± 8.93 | 0.00 ± 0.00 |
| Law School | 82.72 ± 0.19 | 17.53 ± 0.99 | 72.98 ± 0.50 | 0.00 ± 0.00 | 74.36 ± 3.95 | 0.00 ± 0.00 |

Q2: What is the effect of guaranteed fair predictions on performance and overall model fairness? In this experiment, we compare the accuracy of the best baseline model (NNₚ) with CE-Fair predictions on test data. Further, to quantify the effect of fair predictions on model fairness, we introduce flip rate which computes the number of samples in test data where the prediction of the model was modified to satisfy individual fairness. Table 1 demonstrates that you can use CE-Fair predictions to guarantee fairness with accuracy loss up to 8%. This can be explained as follows: since the flip rate of the best-trained model (NNₚ) is high, it is expected that the drop in accuracy has a relation with the model flip rate, as seen with German with the lowest flip rate and smallest decrease in accuracy. Further, we compare our approach against a naive majority-based baseline which is a constant predictor that returns the most frequent label. We observe that, on average, CE-Fair predictions perform 8% better than the majority baseline. In Section 5, we propose a counterexample-guided re-training approach to reduce the performance drop while improving fairness metrics.

Q3: How does CE-FAIR prediction affect inference time? Figure 2 plots the ratio of inference time of NNₚ and CE-Fair predictions for test data on all datasets. We observe that the increase in inference time is proportional to the model flip rate (see Table 1). It also highly depends on the dimension of the sensitive features as seen with IPUMS Adult. This is expected since a larger flip rate means more samples are unfair and therefore more calls to the verification engine. Of course, when violating fairness leads to ethical or legal problems, the question is not whether we can afford fairness enforcement, but whether it is correct to use machine learning at all. In this context, the computational price of enforcing fairness, even if it ends up being significant, is entirely warranted.

5 CE-FAIR TRAINING

In this section, we propose an algorithm for learning fair neural networks with counterexamples. While in the previous section, we guaranteed fair predictions as a post-processing approach, in this section, we propose an in-processing approach that drops the guaranteed fairness requirement in exchange for a relatively fair model with efficient inference. Our learning algorithm is orthogonal to the prediction technique of the previous section and both approaches can be combined to acquire guaranteed fairness with boosted performance and more efficient inference time (see evaluation results in Section 5.1).

To define our learning paradigm, we first define a specific variation of the verification function defined in Definition 3.1 called $\psi_{\max}(f₀, x)$. The CE-Fair verification function can produce a counterexample given an arbitrary sample. To make sure that the counterexamples are representative and not out-of-distribution examples for training, we generate counterexamples relative to each training point. Moreover, to make sure that the counterexamples are effective in reducing the degree of fairness violation, we instead appeal to Definition 5.1 to generate counterexamples with maximal violation relative to each training point.

Definition 5.1. (Maximum Violation CE-Fair Verification) Given a data point $x$ and $f₀ : X \rightarrow [0, 1]$, we find the maximum violation counterexample $x'_{\max}$ using $\psi_{\max}(f₀, x)$ defined as:

$$\arg \max_{x'} |f₀(x') - f₀(x)| \quad s.t. \quad x' = \psi(f₀, x)$$

Suppose a sample data point $x$, is a young single Asian female applicant with a negative decision on her loan application in the training set. Using $\psi_{\max}(f₀, x)$, we are able to find a counterfactual applicant $x'$ with a different combination of sensitive attributes, e.g., an old married American man with the same credit history who can potentially receive a positive decision from the model with the highest level of violation compared to the sample data $x$. Next, we show how we use such counterfactual examples, i.e., counterexamples, to train a fair model.

Definition 5.2. (CE-Fair Training) The loss function for counterexample-guided training can be written as:
Algorithm 2

Input: \( f_0, D, \rho \)

Output: CE-Fair Fine-tuned model: \( f_0 \)

for epoch \( \in \{1 \ldots e\} \) do
  for all \( \text{batch} \in \text{shuffled}(D) \) do
    \( \text{sampling}_\text{batch} \leftarrow \text{RandomSample(batch, } \rho \text{)} \)
    for all \( (x, y) \in \text{sampling}_\text{batch} \) do
      \( \phi_N \leftarrow \text{ModelMIPEncoding}(f_0) \) \{Constraints in Equation 2\}
      \( \phiCE \leftarrow \text{FairCEEncoding}(\phi_N, x, f_0(x)) \) \{Constraints in Equation 3\}
      \( x'_{\text{max}} \leftarrow \text{FindMaxViolationCE}(\phi_N, \phiCE) \) \{Objective as in Equation 8\}
      if \( x'_{\text{max}} \) exists then
        Append(\( \text{sampling}_\text{batch} \), \( (x'_{\text{max}}, y) \))
      end if
    end for
  end for
  \( \theta \leftarrow \text{OptimizationStep}(f_0, \text{sampling}_\text{batch}) \)
end for

Practical Considerations. We highlight the fact that the search for counterexamples is an expensive process and the run time grows with the size of the \( D \), the number of epochs \( e \), and the dimensionality of sensitive features. To make our approach scalable, we introduce a hyperparameter \( \rho \) that indicates what portion of the dataset we are taking. In Section 5.1, we show that even a small value of \( \rho \), as small as 1\%, is effective to fine-tune the model to become fairer in only a few epochs.

Multi-objective Model Selection. In fairness-enforced re-training, we are concerned with accuracy and unfairness at the same time. We thus opt for choosing a Pareto frontier by selecting the epoch whose accuracy and unfairness, when seen as a point in the 2D space, have the minimum \( \varepsilon \) distance to the point corresponding to maximum accuracy and minimum unfairness.

FETA Extension. Extending CE-Fair prediction and training to multi-class classification (one-vs-rest approach) is straightforward. To extend these approaches to regression, e.g., to predict credit limit, we need to modify Equation 3b to encode fairness counterexamples as \( |f_0(x) - f_0(x')| > \varepsilon \) where \( \varepsilon \) is a hyperparameter that needs to be defined based on the context. Also, we need to consider an appropriate loss function, e.g., \( \ell_2 \) loss for regression, to include counterexamples in the training. Note that our neural network MILP encoding is not limited to ReLU and can encode any piece-wise linear activation function.

5.1 Empirical Evaluation of CE-Fair Training

In this section, we evaluate the learning algorithm both on its own and in conjunction with the prediction technique. We use the same datasets and hardware as in Section 4.1.

5.1.1 Experimental Setup. Most of the experimental details are the same as the ones in Section 4.1. Here, we only outline the details specific to CE-Fair Training. As discussed earlier, for CE-Fair training, we have \( \rho = 100\%, 1\%, 2\% \) for German, IPUMS Adult, and Law School, respectively. However, for the final evaluation on the test set (i.e., all the results in the paper), models have been evaluated on the full test set for all datasets.

We re-train the pre-trained \( \text{NN}_b \) model through CE-Fair Training with the same learning rate and batch size used for \( \text{NN}_b \). Each model is CE-Fair trained for 50 epochs with mentioned \( \rho \). Finally, we take the Nadir point of perfect accuracy and perfect \( CE \) rate and take the model from the epoch with minimum \( \varepsilon \) distance to this Nadir point w.r.t. its train metrics.

It is worth mentioning that for CE-Fair Training, the numerical sensitive attributes (like "age") are considered continuous to support a larger space of counterexamples for better regularization.

Q4: Does our CE-Fair training algorithm make the original unfair model fairer? In this experiment, we re-train \( \text{NN}_b \) model with \( \rho = 100\%, 1\%, 2\% \) for German, IPUMS Adult, and Law School, respectively. We train for 50 epochs and select the best model based on the Pareto frontier discussed in Section 5. Figure 3 summarizes the learning curves. We observe that while loss oscillates due to the fairness-performance tradeoff, the average of maximum violation substantially decreases. The results presented here are chosen among the \text{FULL BATCH} and \text{CE BATCH} options. CE BATCH decreases average violation almost to its minimum only in the first few epochs. This is because it is focusing only on the counterexamples while \text{FULL BATCH} experiences a more smooth curve.
Table 2: The effect of CE-Fair re-training on fairness metrics – 5-fold CV results

| Dataset    | Approach       | Accuracy | Flip Rate | CE Rate |
|------------|----------------|----------|-----------|---------|
| German     | NN CE-Fair     | 76.70 ± 2.78 | 8.90 ± 0.73 | 29.80 ± 2.15 |
|            | CE-Fair Training | 75.70 ± 3.77 | 3.60 ± 0.73 | 17.70 ± 4.67 |
| IPUMS Adult| NN CE-Fair     | 81.57 ± 0.47 | 24.39 ± 1.52 | 89.40 ± 1.80 |
|            | CE-Fair Training | 80.03 ± 0.57 | 2.80 ± 0.67 | 15.34 ± 2.53 |
| Law School | NN CE-Fair     | 82.72 ± 0.19 | 17.53 ± 0.59 | 41.18 ± 2.05 |
|            | CE-Fair Training | 84.99 ± 0.28 | 5.06 ± 0.97 | 7.64 ± 0.89 |

To quantify if the function is fairer, we compare two fairness metrics: Flip Rate and CE Rate, defined in Section 4.1. As shown in Table 2, counterexample-guided retraining leads to better fairness metrics on all datasets by reducing the number of fairness violations. In fact, in the IPUMS Adult dataset, we see the highest decrease of 74% w.r.t. CE Rate. These results indicate the usefulness of CE-Fair learning to make the original unfair model fairer. Further, the drop in accuracy when enforcing fairness is negligible when compared to the original model. In fact, we observe an increase in accuracy for Law School. While CE-Fair training significantly reduces fairness violations, it does not guarantee fair predictions for all points in the input domain. This motivates the need for using CE-Fair predictions in conjunction with the counterexample-guided learning algorithm, to guarantee fair predictions.

Table 3: Comparison of applying CE-Fair Prediction on NN$_b$ vs. on the CE-Fair re-trained model – 5-fold CV results

| Dataset    | Approach       | Accuracy | Inference Time (s) |
|------------|----------------|----------|--------------------|
| German     | CE-Fair Prediction | 74.20 ± 1.50 | 0.45 ± 0.06       |
|            | CE-Fair Training + CE-Fair Prediction | 75.30 ± 3.65 | 0.39 ± 0.03       |
| IPUMS Adult| CE-Fair Prediction | 73.23 ± 0.93 | 119.44 ± 33.45    |
|            | CE-Fair Training + CE-Fair Prediction | 74.06 ± 0.71 | 10.20 ± 2.90      |
| Law School | CE-Fair Prediction | 74.36 ± 1.95 | 0.39 ± 0.14       |
|            | CE-Fair Training + CE-Fair Prediction | 84.14 ± 0.97 | 0.30 ± 0.13       |

Q5: Does counterexample-guided learning improve the quality of the guaranteed prediction model? As shown in Table 3, by additionally enforcing fairness constraints through counterexample-guided re-training, we improve both accuracy and inference time of CE-Fair predictions. Running CE-Fair predictions on the re-trained model improves inference runtime significantly on IPUMS Adult which has the largest sensitive feature space. The maximum drop in accuracy compared to NN$_b$ is only 1.5%. Whereas, running CE-Fair predictions directly on NN$_b$ leads to a maximum accuracy loss of 8.3%. Thus, with CE-Fair training, we get both a fairness guarantee and better runtime and model performance.

Q6: How does FETA perform compared to fairness under unaware model?

We train a model that is unaware of the sensitive features. Such a model would satisfy fairness definition 2.1 as its decision is not prone to changes in the sensitive attributes. We call this model the blind model.

Table 4: Accuracy of different fair models compared – 5-fold CV

| Dataset    | Blind Model | CE-Fair Training + Prediction |
|------------|-------------|------------------------------|
| German     | 70.30 ± 2.20 | 75.30 ± 3.65 |
| IPUMS Adult| 78.90 ± 0.15 | 79.46 ± 0.71 |
| Law School | 74.19 ± 1.20 | 84.14 ± 0.97 |

In Table 4, we compare the blind model with the CE-Fair Training + Prediction models. We train the blind model for the same amount of epochs and the same $\rho$ (ratio) as CE-Fair Training for a fair comparison. Note that the blind model is fair by design but similar to CE-Fair Training, it requires re-training the model. We observe that CE-Fair Training + Prediction, aka, the FETA pipeline, gives consistently better accuracy, up to 10% better compared to the blind model, while providing similar fairness guarantees.

Q7: How does FETA compare against existing work?

The literature on fairness in machine learning contains several well-established notions, particularly group fairness notions like demographic parity, equalized odds, and equal opportunity [24]. However, as previously demonstrated [8], it is not straightforward to achieve both group fairness and individual fairness in a single model. To compare our framework with group fairness mitigation techniques, an extension to group fairness notions would be required, but that falls outside the scope of our current work. Table 5 reports the accuracy and CE Rate of FETA compared to a recent method called LCIFR [54] that mitigates individual fairness using the same fairness definition as ours (i.e., Definition 2.1). To the best
of our knowledge, LCIFR is the closest related work to ours that includes the same fairness notion. In LCIFR [54], the authors propose a fair representation learning approach and adversarial classification to address individual fairness. We fine-tune both methods on all datasets and extend LCIFR to accept multiple sensitive attributes similar to our setting. To gather the results of this section, we extend the sensitive features of LCIFR and use our own train/test split, we also report the empirical results using the original implementation in the Appendix. We set \( \gamma = 1.0 \) (their loss balancing factor) for the sake of fair comparison. The results in Table 5 indicate that the accuracy of FETA outperforms LCIFR on German and Law School, and only for IPUMS Adult, LCIFR trains a more accurate model. This finding is noteworthy because LCIFR combines both pre-processing and in-processing techniques to enhance performance while our approach focuses on in-processing and post-processing techniques. It is worth saying that our approach guarantees to have no CE rate for all three datasets while LCIFR cannot provide any guarantees, i.e., LCIFR results are empirical and even if they exhibit close to zero unfairness on the test set, it does not imply guaranteed fairness over all the data points in the input space. Also, we use a combination of categorical and continuous sensitive features while for LCIFR we only use the categorical ones due to their design. Note that using continuous sensitive features results in a larger counterexample space.

| Dataset  | Approach      | Accuracy (%) | CE Rate (%) |
|----------|---------------|--------------|-------------|
| German   | NN            | 76.70 ± 2.78 | 29.80 ± 2.15 |
|          | LCIFR [54]    | 72.30 ± 1.43 | 0.20 ± 0.24  |
|          | FETA(CE-Fair Train. + Pred.) | 75.30 ± 3.65 | 0.9 ± 0.0    |
| IPUMS Adult | NN            | 81.57 ± 0.47 | 89.40 ± 1.80 |
|          | LCIFR [54]    | 81.52 ± 0.34 | 0.05 ± 0.04  |
|          | FETA(CE-Fair Train. + Pred.) | 79.46 ± 0.71 | 0.0 ± 0.0    |
| Law School | NN            | 82.72 ± 0.19 | 41.18 ± 2.05 |
|          | LCIFR [54]    | 74.13 ± 0.76 | 0.008 ± 0.007|
|          | FETA(CE-Fair Train. + Pred.) | 84.14 ± 0.97 | 0.0 ± 0.0    |

### 6 EXTENSIONS TO FETA

We would like to highlight the fact that any extensions to FETA that are expressible as MILP constraints are straightforward. The goal of this work is to present a working pipeline of FETA that results in a provable post-processed model with high utility and guaranteed fairness. Within this framework, one can take the liberty to leverage the flexibility of MILP for straightforward extensions to FETA in different directions, as highlighted by the examples below.

- **Model architecture.** While our study focuses on ReLU NNs, it is feasible to extend our approach to other piece-wise linear activation functions such as Leaky ReLU and the PReLU (Parametric ReLU). Moreover, there are several non-linear activation functions that are not piece-wise linear, but they can be approximated using piece-wise linear functions. To give an example, the sigmoid function is a smooth, S-shaped curve that can be approximated using piece-wise linear functions [6]. Similarly, the hyperbolic tangent (tanh) function, which is another commonly used activation function in neural networks, can be approximated using piece-wise linear functions [55]. Finally, the softmax function which is often used as the activation function in the output layer of neural networks for multi-class classification tasks, can be approximated using quantiles. In this method, the output range of the Softmax function is divided into several intervals or quantiles. Each interval is then approximated by a linear function that passes through two points: the lower and upper bound of the interval. Although approximating non-linear activation functions with piece-wise linear functions can simplify the computations involved in neural network training and inference, such approximations would result in losing the post-processing guarantees presented in this work. Finally, extension to more diverse architectures that are suitable for non-tabular data, including max pooling or batch normalization layers, is also theoretically straightforward, however, it would incur extra computational costs.

- **Fairness notion and similarity measure.** One will face many options to define similarity for individual fairness. Examples include an \( \ell_1 \) distance with a threshold \( \epsilon \) or even a separate NN trained to output a similarity; both may be encoded as MILP constraints.

  In this paper, we evaluate FETA using the casual discrimination notion of individual fairness as defined in [19] (see Definition 2.1). However, extending FETA to leverage other notions of individual fairness is feasible. A simple extension to our fairness notion is \( \epsilon \)-individual fairness which indicates that for any two individuals \( x \) and \( x' \) whose feature vectors differ by at most \( \epsilon \), their model outputs should be the same. This can be encoded as a piece-wise linear constraint using the \( \ell_1 \) distance with a threshold \( \epsilon \), which can be encoded easily as MILP constraints.

Another plausible extension is to consider fairness notions that are capable of capturing relations among features. A notable example is counterfactual fairness [39] which is a causal fairness measure. This notion requires that the model’s predictions should not depend on an individual’s protected attributes (e.g., race or gender) except through the attributes that are causally related to the outcome. Counterfactuals are often defined within Pearl’s Structural Causal Model (SCM) framework [50] to capture relations among features. This framework defines a causal model by a set of so-called structural equations. If one defines the structural equations in a piece-wise linear format (see [40], counterfactual fairness metric can be easily encoded in FETA. However, it is important to acknowledge that constructing an SCM for counterfactual fairness requires carefully identification of relevant variables and their causal relationships, precise specification of functional relationships, and simulation of various scenarios to evaluate fairness. This is a complex undertaking that demands specialized expertise.

Finally, another approach is to train a separate neural network on a population data to output a similarity score between two individuals given the features in the input space of the model. The output of the neural network can then be discretized into several intervals, and a separate linear function can be used to approximate the similarity within each interval, which can then encoded as MILP constraints.
• **Expert constraints.** Another way to expand FETA is by incorporating constraints that capture domain knowledge. For example, constraints can be defined on how features are permitted to change when searching for a counterexample. As an example, in FETA we make sure that a categorical feature \( H \) with \( k \) values remains as 1-hot encoding by imposing constraints such as \( \sum_{i=1}^{k} h_i = 1 \). Or in the credit card scenario, a constraint can be added to ensure that credit history is always less than age.

7 RELATED WORK

Our paper is a contribution to the extensive literature on fairness in machine learning. In this section, we will contextualize our work and compare it to existing techniques for mitigating and verifying fairness in machine learning.

**Bias Mitigation Algorithms.** Methods that seek to introduce fairness into machine learning systems broadly fall into one of three categories: pre-processing, in-processing, and post-processing. The pre-processing approaches try to transform training data [11, 32] to mitigate bias. E.g., the most related work to ours is a fair representation learning technique by [41, 54] that uses the same fairness notion as we used in our paper (see Definition 2.1) to learn a fair representation of the individuals and train a certified machine learning model that accepts the fair representation as input. In-processing techniques mitigate discrimination via model regularization by directly modifying the learning algorithm to meet the fairness criteria. The regularization implicitly or explicitly optimizes a fairness metric. Several in-processing techniques have been proposed for group-fairness metrics [25, 35, 46]. Further, in-processing techniques for individual fairness fall into two categories based on whether they enforce fairness with or without access to individual fairness metrics. The first category of work circumvents the need for a fair metric by assuming the learner has access to an oracle that provides feedback on violations of individual fairness [20, 31]. The second category of work enforces fairness by assuming access to a fairness metric and using adversarial learning techniques to enforce fairness [67, 68]. Post-processing techniques modify the model’s prediction during the inference time to make sure that the prediction distribution approaches a specific fairness metric [1, 24, 44, 51, 52].

In this paper, we propose both in-processing and post-processing techniques for individual fairness. Unlike prior work, our proposed in-processing technique does not depend on oracles for feedback on violations [20, 31] and the fair training algorithm uses counterexamples rather than adversarial training [67, 68] to enforce fairness. Further, adversarial training is approximate adversarial examples, whereas we find exact counterexamples. While we are similar to [7, 36] in terms of in-processing, we propose a post-processing approach that, unlike others, guarantees individual fairness. Moreover, our in-processing technique is only a help to improve the quality of the ultimate post-processed model with guarantees. While most of the existing work on post-processing focus on group fairness or other definitions of individual fairness [1, 24, 44, 51, 52], we propose guaranteed predictions via post-processing using the individual fairness definition from [19]. Although Post-processing by [44] employs the same definition of individual fairness as our method, it only supports a single binary sensitive attribute. Moreover, their method works by training a new model on the predictions of the original model, with the goal of transforming the predictions to be more fair. The new model is trained to optimize a fairness objective, such as minimizing the distance between the predicted outcomes of similar individuals with different sensitive attributes. In contrast, our approach can handle multiple binary/continuous sensitive attributes and do not rely on a trained model to generalize at the prediction time and provide formal provable guarantees of fair predictions for all points in the domain. This makes our approach more versatile and robust, with the ability to accommodate a wider range of real-world scenarios while ensuring fairness.

**Verifying machine learning systems and Adversarial Learning.** Prior work on machine learning verification can be classified into (i) verification using satisfiability modulo theory (SMT) or mixed-integer linear programming (MILP) [34, 61], and (ii) verification using convex relaxations [16, 56]. Our proposed approach uses the MILP encoding from prior work [47] to build a practical individual fairness verification approach. Further, prior works on fairness verification have been proposed in the context of probabilistic programs [2, 3] or linear kernels [29]. Recent works propose adversarially robust algorithms which can be divided into empirical [38, 45] and certified defenses [26, 53, 57, 64]. Specifically [48, 65, 68] propose adversarial training-based algorithms for fairness. We are closely related to these works, in that we carry out adversarial training using counterexamples. However, we differ in two ways. First, to the best of our knowledge, there is no related work in the adversarial robustness literature for ensuring individual fairness using the definition from [19]. Second, related work in adversarial training only ensures correctness in the neighbourhood of a training point, while we globally search for a counterexample and are able to discover long-range fairness violations.

8 CONCLUSION & FUTURE DIRECTIONS

In this work, we propose 1) a counterexample-guided fairness verification framework, 2) a counterexample-guided approach to guarantee fairness as a post-processing approach without intervening in the model, 3) a counterexample-guided approach to adapt an already trained model toward being fair, which cannot guarantee fairness on its own but can be combined with the former approach to provide better accuracy and faster inference, 4) An open-source tool called FETA that facilitates the integration of multiple techniques for optimal results. We showed using real-world datasets that in practice we can have efficient and fair models with little damage to accuracy. While the results of our approaches are promising, we note that the causal discrimination fairness notion adopted in this work is limited; it is important to recognize that features may be interrelated in certain domains and that our concept of fairness has limitations. However, we must also stress the complexity of achieving individual fairness in both the training and post-processing stages. It should be noted that ensuring individual fairness means ensuring that the model does not discriminate between any two similar pairs of individuals in the outcome space. This is distinct from placing constraints on the parameters of the trained model during the training process. An interesting future work would be to extend FETA with other fairness notions that are capable of capturing the relations among features. A notable example is
counterfactual fairness [39] which is a causal fairness measure based on SCM (structural causal model). Another future avenue to explore is to bind the counterexamples to follow the distribution of the data. One can extend our framework by adding distribution constraints to our MILP formulation to restrict counterexamples to be Out-of-Distribution. Another limitation of FETAleft for future work which is typical of approaches where such guarantees are provided, is scalability. Every year, state-of-the-art neural networks grow in size with a large number of parameters which poses incredible challenges for constraint-based verification approaches. Although the neural network model used in our experiments is fairly complex for tabular data, this approach might not scale to very deep networks. Indeed, it would be an interesting direction to address and explore the limits of scaling the model architecture. For example, future work might study how to modify the neural network learning algorithms to enable scalable constraint-based analysis. Finally, solvers tend to use floating-point approximations leading to numeric instabilities. Problem-specific solutions to make the approach more numerically stable could be a potential future work as well.

9 SOCIETAL IMPACT

In this paper, we propose three approaches to enhance the fairness of neural network models by fairness verification, fair training, and fair prediction. The impact of using these approaches to have fair neural network models on society is considerable as they prevent these models from incurring unfair biases or discrimination against individuals. Having access to fair models can enhance the accessibility of resources, opportunities, and services for historically marginalized people, such as people of color, women, people with disabilities, and low-income individuals.

In addition, individually fair neural network models can enhance trust and transparency in decision-making systems, particularly in critical areas such as criminal justice, hiring, and lending, where biased or unfair decisions can severely impact individuals.

Moreover, fair neural network models can help promote diversity and inclusion. By recognizing the importance of fairness and individual fairness in their models, companies can attract a more diverse group of users and employees, which can lead to better products and services for everyone.

However, while fair neural network models have the potential to promote fairness and reduce discrimination, it is important to carefully consider their potential drawbacks and limitations. In this paper, we focus on a specific fairness notion to verify, train and guarantee fair prediction of neural network models. However, we acknowledge that there is a huge literature on various notions of fairness, and individual fairness is context-dependent and should be defined relative to a task. Hence, our framework cannot and should not be used in every application domain.

Furthermore, there is a risk that fair neural network models could be misused or misinterpreted. One potential disadvantage of fair neural network models is that they may not always be able to achieve perfect fairness. It is worth mentioning that many aspects of fairness are not captured by mathematical measures. Our framework is highly dependent on a fairness notion, and the result change by changing the notion of fairness. Although individual fairness can be defined in different ways in FETA as explained in Section 6, it is necessary to choose only one definition for deployment; various individual fairness metrics may conflict with each other, making it difficult to optimize for all of them simultaneously.

Finally, although our approach can produce fair predictions, it is still based on a model produced by a machine learning algorithm. And it is important to note that FETA could suffer from the same disadvantages as the original model in aspects that we did not consider in this work, such as privacy, explanation, safety, security, and robustness. Hence, the user must be aware of such a system’s limitations, especially when using these models to replace people in decision-making.

REFERENCES

[1] Aniya Agarwal, Pranay Loia, Seema Nagar, Kuntal Dey, and Diptikalyan Saha. 2018. Automated test generation to detect individual discrimination in AI models. arXiv preprint arXiv:1809.05260 (2018).
[2] Aws Albarghouthi, Loris D’Antoni, Samuel Drews, and Aditya V Nori. 2017. Fairsquare: probabilistic verification of program fairness. Proceedings of the ACM on Programming Languages 1, OOPSLA (2017), 1–30.
[3] Oubert Bastani, Xin Zhang, and Armando Solar-Lezama. 2019. Probabilistic verification of fairness properties via concentration. Proceedings of the ACM on Programming Languages 3, OOPSLA (2019), 1–27.
[4] BBC. 2018. Gender pay gap: Men still earn more than women at most firms. https://www.bbc.com/news/business-43129393
[5] BBC. 2019. Apple’s ‘sexist’ credit card investigated by US regulator. https://www.bbc.com/news/business-50365609
[6] Valeria Beiu, Jan A Peperstraete, Joes Vandewalle, and Rudy Lauwereins. 1994. VLSI complexity reduction by piece-wise approximation of the sigmoid function. In ESANN. Citeseer.
[7] Elias Benussi, Andrea Fatane, Matthew Wicker, Luca Laurenti, and Marta Kwiatkowska. 2022. Individual Fairness Guarantees for Neural Networks. In Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, JCAI-22, Lud De Raedt (Ed.). International Joint Conferences on Artificial Intelligence Organization, 651–658. https://doi.org/10.24963/ijcai.2022/92 Main Track.
[8] Reuben Binns. 2020. On the apparent conflict between individual and group fairness. In Proceedings of the 2020 conference on fairness, accountability, and transparency. 514–524.
[9] Rudy Bunel, Ilker Turkenaus, Philip H.S. Torr, Pushmeet Kohli, and M. Pawan Kumar. 2018. A Unified View of Piecewise Linear Neural Network Verification. In Proceedings of the 32nd International Conference on Neural Information Processing Systems (Montréal, Canada) (NIPS’18). Curran Associates Inc., Red Hook, NY, USA, 4795–4804.
[10] Joy Buolamwini and Timnit Gebru. 2018. Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification. In Proceedings of the 1st Conference on Fairness, Accountability and Transparency (Proceedings of Machine Learning Research, Vol. 81). Sorelle A Friedler and Christo Wilson (Eds.). PMLR, 77–91. https://proceedings.mlr.press/v81/buolamwini18a.html
[11] Flavio Calmon, Dennis Wei, Bhamukiran Vinzamuri, Karthikeyan Natesan Raman- murthy, and Kush R Varshney. 2017. Optimized pre-processing for discrimination prevention. Advances in neural information processing systems 30 (2017).
[12] Yooljung Choi, Golnoosh Farradi, Behrouz Babaki, and Gyu Van den Boeck. 2020. Learning fair naive bayes classifiers by discovering and eliminating discriminating patterns. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34. 10077–10084.
[13] Edmund Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. 2000. Counterexample-guided abstraction refinement. In International Conference on Computer Aided Verification. Springer, 154–169.
[14] Frances Ding, Moritz Hardt, John Miller, and Ludwig Schmidt. 2021. Retiring Adult: New Datasets for Fair Machine Learning. In Advances in Neural Information Processing Systems, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34. Curran Associates, Inc., 6478–6490. https://proceedings.neurips.cc/paper/2021/file/3e254441e6382a7fbc6bba3e45059-Paper.pdf
[15] Marcelo Carvalho dos Santos, Victor Henrique Cabral Pinheiro, Filipe Santanta Moreira do Desterro, Renato Koga de Avellar, Roberto Schirru, Andressa dos Santos Nicolau, and Alan Miranda Monteiro de Lima. 2019. Deep rectifier neural network applied to the accident identification problem in a PWR nuclear power plant. Annals of Nuclear Energy 133 (2019), 400–408.
[16] Krishnamurthy Dvijotham, Robert Stanforth, Sven Gowal, Timothy A Mann, and Pushmeet Kohli. 2018. A Dual Approach to Scalable Verification of Deep
