CertiFair: A Framework for Certified Global Fairness of Neural Networks

Haitham Khedr, Yasser Shoukry
University of California, Irvine
{hkhedr, yshoukry}@uci.edu

Abstract

We consider the problem of whether a Neural Network (NN) model satisfies global individual fairness. Individual Fairness (defined in (Dwork et al. 2012)) suggests that *similar* individuals with respect to a certain task are to be treated *similarly* by the decision model. In this work, we have two main objectives. The first is to construct a verifier which checks whether the fairness property holds for a given NN in a classification task or provides a counterexample if it is violated, i.e., the model is fair if all similar individuals are classified the same, and unfair if a pair of similar individuals are classified differently. To that end, we construct a sound and complete verifier that verifies global individual fairness properties of ReLU NN classifiers using distance-based similarity metrics. The second objective of this paper is to provide a method for training provably fair NN classifiers from unfair (biased) data. We propose a fairness loss that can be used during training to enforce fair outcomes for similar individuals. We then provide provable bounds on the fairness of the resulting NN. We run experiments on commonly used fairness datasets that are publicly available and we show that global individual fairness can be improved by 96 % without a significant drop in test accuracy.

1 Introduction

Neural Networks (NNs) have become an increasingly central component of modern decision-making systems, including those that are used in sensitive/legal domains such as crime prediction (Brennan, Dieterich, and Ehret 2009), credit assessment (Dua and Graff 2017), income prediction (Dua and Graff 2017), and hiring decisions. However, studies have shown that these systems are prone to biases (Mehrabi et al. 2021b) that deem their usage *unfair* to unprivileged users based on their age, race, or gender. The bias is usually either inherent in the training data or introduced during the training process. Mitigating algorithmic bias has been studied in the literature (Zhang, Lemoine, and Mitchell 2018; Xu et al. 2018; Mehrabi et al. 2021a) in the context of group and individual fairness. However, the fairness of the NN is considered only *empirically* on the test data with the hope that it represents the underlying data distribution.

Unlike the *empirical* techniques for fairness, we are interested to provide *provable certificates* regarding the fairness of a NN classifier. In particular, we focus on the "global individual fairness" property which states that a NN classification model is globally individually fair if *all similar* pairs of inputs x, x' are assigned the same class. We use a feature-wise closeness metric instead of an ℓ_p norm to evaluate similarity between individuals, i.e, a pair x, x' is similar if for all features i, $|x_i - x'_i| \leq \delta_i$. Given this fairness notion, the objective of this paper is twofold. First, it aims to provide a sound and complete formal verification framework that can automatically certify whether a NN satisfies the fairness property or produces a concrete counterexample showing two inputs that are not treated fairly by the NN. Second, this paper provides a training procedure for certified fair training of NNs even when the training data is biased.

Challenge: Several existing techniques focus on generalizing ideas from *adversarial robustness* to reason about NN fairness (Yurochkin, Bower, and Sun 2020; Ruoss et al. 2020). By viewing unfairness as an adversarial noise that can flip the output of a classifier, these techniques can certify the fairness of a NN *locally*, i.e., in the neighborhood of a given individual input. In contrast, this paper focuses on *global* fairness properties where the goal is to ensure that the NN is fair with respect to *all* the similar inputs in its domain. Such a fundamental difference precludes the use of existing techniques from the literature on adversarial robustness and calls for novel techniques that can provide provable fairness guarantees.

This work: We introduce CertiFair, a framework for certified global fairness of NNs. This framework consists of two components. First, a verifier that can prove whether the NN satisfies the fairness property or produce a concrete counterexample that violates the fairness property. This verifier is motivated by the recent results in the "relational verification" problem (Barthe, Crespo, and Kunz 2011) where the goal is to verify *hyperproperties* that are defined over pairs of program traces. Our approach is based on the observation that the global individual fairness property (1) can be seen as a *hyperproperty* and hence we can generalize the concept of *product programs* to *product NNs* that accepts a pair of inputs (x, x'), instead of a single input x, and generates two independent outputs for each input. A global fairness property for this product NN can then be verified using existing NN verifiers. Moreover, inspired by methods in certified robustness, we also propose a training procedure for *certified fairness* of NNs. Thanks again to the product NN, mentioned above, one can establish upper bounds on fairness and use it as a regularizer during the training of NNs. Such a regularizer will promote the fairness of the resulting model, even if the data used for training is biased and can lead to an unfair classifier. While such *fairness regularizer* will enhance the fairness of the model, one needs to check if the fairness property holds globally using the sound and complete verifier mentioned above.

Contributions: Our main contributions are:

- We present a sound and complete NN verifier for global individual fairness properties.
- A method for training NN classifiers with a modified loss that enforces fair outcomes for similar individuals. We provide bounds on the loss in fairness which constructs a certificate on the fairness of the trained NN.
- We applied our framework to common fairness datasets and we show that global fairness can be achieved with a minimal loss in performance.

2 Preliminaries

Our framework supports regression and multi-class classification models, however for simplicity, we only present our framework for binary classification models $h : \mathbb{R}^n \to \{0,1\}$ of the form $h(x) = t(f_\theta(x))$ where t is a threshold function with threshold equals to 0.5. Moreover, we assume f_θ is an L-layer NN with ReLU hidden activations and parameters $\theta = ((W_1, b_1), \dots, (W_L, b_L))$ where (W_i, b_i) denotes the weights and bias of the ith layer. We also assume the activation function of the last layer of f_θ is a sigmoid function. The NN accepts an input vector x where the components $x_i \in \mathbb{R}$ (the set of real numbers) or $x_i \in \mathbb{Z}$ (the set of integer numbers). This is suitable for most datasets where some features of the input are numerical while others are categorical[1]. In this paper, we are interested in the following fairness property:

Definition 2.1 (Global Individual Fairness (Dwork et al. 2012; John, Vijaykeerthy, and Saha 2020))**.** A model $f_\theta(x)$ is said to satisfy the global individual fairness property ϕ if the following holds:

$$\forall x, x' \in D_\phi \quad \text{s.t.} \quad d(x, x') = 1 \implies h(f_\theta(x)) = h(f_\theta(x')) \tag{1}$$

where $d : \mathcal{R}^n \times \mathcal{R}^n \to \{1, 0\}$ is a similarity metric that evaluates to 1 when x and x' are similar and D_ϕ is the input domain of x for property ϕ defined as $D_\phi := D_\phi^0 \times ... \times D_\phi^{n-1}$ with $D_\phi^i := \{x_i \mid l_i \le x_i \le u_i\}$ for some bounds l_i and u_i.

In this paper, we utilize the feature-wise similarity metric d defined as:

$$d(x, x') = \begin{cases} 1 & \text{if } |x_i - x_i'| \le \delta_i \qquad \forall i \in \{1, \dots n\} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

[1]More specifically, categorical inputs are one-hot encoded and our framework splits the input space into partitions, each with a constant value of the one-hot encoded vector. More details are explained in Section 5).

This feature-wise similarity metric allows the fairness property ϕ in (1) to capture several other fairness properties as special cases as follows:

Definition 2.2 (Individual discrimination (Aggarwal et al. 2019))**.** A model $f_\theta(x)$ is said to be nondiscriminatory between individuals if the following holds:

$$\forall x = (x_s, x_{ns}), x' = (x_s', x_{ns}') \in D_\phi$$
$$\text{s.t.} \quad x_{ns} = x_{ns}' \text{ and } x_s \neq x_s' \implies h(f_\theta(x)) = h(f_\theta(x')),$$

where x_s and x_{ns} denotes the sensitive attributes and non-sensitive attributes of x, respectively.

Indeed, the individual discrimination corresponds to a global individual fairness property by setting $\delta_i = 0$ in (2) for the non-sensitive attributes and $\delta_i = u_i - l_i$ for sensitive attributes. Another definition of fairness (Ruoss et al. 2020) states that two individuals are similar if their numerical features differ by no more than α. Again, this can be represented by the closeness metric simply by setting $\delta_i = 0$ for categorical attributes and $\delta_i = \alpha$ for numerical attributes.

Based on Definition 2.1, we can formally verify whether the fairness property ϕ holds by checking if the set of counterexamples (or violations) $\mathcal{C}$ is empty, where $\mathcal{C}$ is:

$$\mathcal{C} = \left\{ (x, x') \middle| x, x' \in D_\phi, \bigwedge_{i=0}^{n-1} |x_i - x_i'| < \delta_i, h(x) \neq h(x') \right\}. \tag{3}$$

3 Global Individual Fairness as a Hyperproperty

In this section, we draw the connection between the verification of global individual fairness properties (1) and hyperproperties in the context of program verification. On the one hand, several local properties of NNs (e.g., adversarial robustness) are considered trace properties, i.e., properties defined on the input-output behavior of the NN. In this case, one can search the input space of the NN to find a *single* input (or counterexample) that leads to an output that violates the property. In the domain of adversarial robustness, a counterexample corresponds to a disturbance to an input that can change the classification output of a NN. On the other hand, other properties, like the global fairness properties, can not be modeled as trace properties. This stems from the fact that one can not judge the correctness of the NN by considering individual inputs to the NN. Instead, finding a counterexample to the fairness property will entail searching over *pairs* of inputs and comparing the NN outputs of these inputs. Properties that require examining pairs or sets of traces (input-outputs of a program) are defined as *hyperproperties* (Barthe, Crespo, and Kunz 2011).

Modeling global individual fairness as a hyperproperty leads to a direct certification framework. In particular, a key idea in the hyperproperty verification literature is the notion of a product program that allows the reduction of the hyperproperty verification problem to a standard verification problem (Barthe, Crespo, and Kunz 2011). A product program is constructed by composing two copies of the original program together. The main benefit is that the hyperproperties

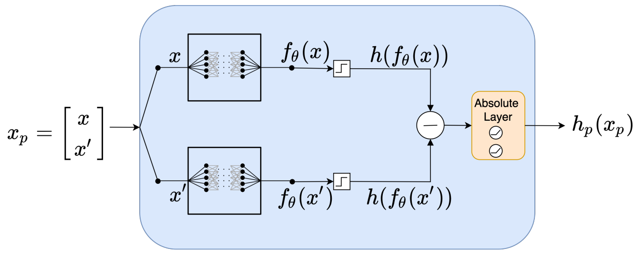

Figure 1: Construction of the Product NN.

of the original program become trace properties of the product program that can be verified using standard techniques. Motivated by this observation, our framework CertiFair generalizes the concept of *product programs* into *product NNs* (described in detail in Section 4.2 and shown in Figure 1) that accepts a pair of inputs and generates a pair of two independent outputs. We then use the product network to verify fairness (hyper)properties using standard techniques.

4 CertiFair: A Framework for Certified Fairness of Neural Networks

As mentioned earlier in section 3, the fairness property can be viewed as a hyperproperty of the NN. We propose the use of a product NN that can reduce the verification of such hyperproperty into standard trace (input/output) property. In this section, we first explain how to construct the product NN followed by how to use it to encode the fairness verification problem into ones that are accepted by off-the-shelf NN verifiers. Next, we discuss how to use this product NN to derive a fairness regualrizer that can be used during training to obtain a certified fair NN.

4.1 Product Neural Network

Given a neural network f_θ, the product network f_{θ_p} is basically a side-to-side composition of f_θ with itself. More formally, the parameters vector θ_p of the product NN is defined as:

$$\theta_p = \left(\left(\begin{bmatrix} W_1 & \mathbf{0} \\ \mathbf{0} & W_1 \end{bmatrix}, \begin{bmatrix} b_1 \\ b_1 \end{bmatrix} \right), \ldots, \left(\begin{bmatrix} W_L & \mathbf{0} \\ \mathbf{0} & W_L \end{bmatrix}, \begin{bmatrix} b_L \\ b_L \end{bmatrix} \right) \right)$$

where (W_i, b_i) are the weights and biases of the ith layer of f_θ. The input to the product network f_{θ_p} is a pair of concatenated inputs $x_p = (x, x')$. Finally, we add an output layer that results in an output $h_p \in \{0, 1\}$ defined as: $h_p(x_p) = |h(f_\theta(x)) - h(f_\theta(x'))|$ where the absolute value operator $|.|$ can be implemented using ReLU nodes by noticing that $|a| = max(a, 0) + max(-a, 0)$. Figure 1 summarizes this construction.

4.2 Fairness Verification

Using the product network defined above, we can rewrite the set of counterexamples $\mathcal{C}$ in (3) as:

$$\mathcal{C}_p = \left\{ x_p \middle| x_p \in D_\phi \times D_\phi,\ \bigwedge_{i=0}^{n-1} |x_i - x'_i| < \delta_i,\ h_p(x_p) > 0 \right\} \quad (4)$$

which corresponds to the standard verification of NN input-output properties (Liu et al. 2021), albeit being defined over the product network inputs and outputs.

To check the emptiness of the set $\mathcal{C}_p$ in (4) (and hence certify the global individual fairness property), we need to search the space $D_\phi \times D_\phi$ to find at least one counterexample that violates the fairness property, i.e., a pair $x_p = (x, x')$ that represent similar individuals who are classified differently by the NN. Finding such a counterexample is, in general, NP-hard (Katz et al. 2017) due to the non-convexity of the ReLU NN f_{θ_p}. To that end, we use PeregriNN (Khedr, Ferlez, and Shoukry 2021) as our NN verifier. Briefly, PeregriNN overapproximates the highly non-convex NN with a linear convex relaxation for each ReLU activation. This is done by introducing two optimization variables for each ReLU, a pre-activation variable $\hat{y}$ and a post-activation variable y. The non-convex ReLU function can then be overapproximated by a triangular region of three constraints; $y \geq 0$, $y \geq \hat{y}$, and $y \leq \frac{u}{u-l}(\hat{y} - l)$, where l,u are the lower and upper bounds of $\hat{y}$ respectively. The solver tries to check whether the approximate problem has no solution or iteratively refines the NN approximation until a counterexample that violates the fairness constraint is found. PeregriNN employs other optimizations in the objective function to guide the refinement of the NN approximation but the details of these methods are beyond the scope of this paper. We refer the reader to the original paper (Khedr, Ferlez, and Shoukry 2021) for more details on the internals of the solver.

Proposition 4.1. *Consider a NN model f_θ and a fairness property ϕ—either representing a Global Individual Fairness property (Definition 2.1) or an Individual Discrimination property (Definition 2.2). Consider a set of counterexamples $\mathcal{C}_p$ computed using a NN verifier applied to the product network f_{θ_p}. The NN satisfies the property ϕ whenever the set $\mathcal{C}_p$ is empty.*

Proof. This result follows directly from the equivalence between the sets $\mathcal{C}$ in (3) and $\mathcal{C}_p$ in (4) along with the NN verifiers (like PeregriNN) being sound and complete and hence capable of finding any counterexample if one exists. □

4.3 Certified Fair Training

In this section, we formalize a *fairness regularizer* that can be used to train certified fair models. In particular, we propose two fairness regularizers that correspond to *local* and *global* individual fairness. We provide the formal definitions of both these regularizers below and their characteristics.

Local Fairness Regularizer $\mathcal{L}^l_f$ using Robustness around Training Data: Our first proposed regularizer is motivated by the *robustness* regularizers used in the literature of certified robustness (Wong and Kolter 2018; Zhang et al. 2019). The regularizer, denoted by $\mathcal{L}^l_f$, aims to minimize the average loss in fairness across all training data. More formally, given a training point (x, y) and NN parameters θ, let $\mathcal{L}(f_\theta(x), y; \theta) = -[y \log(f_\theta(x)) + (1 - y) \log(1 - f_\theta(x))]$ be the standard binary cross-entropy loss. The fairness regularizer $\mathcal{L}^l_f$ can then be defined as:

$$\mathcal{L}_f^l(\theta) = \mathop{\mathbb{E}}_{(x,y)\in(X,Y)}\left[\max_{\substack{x'\in D_\phi\\ d(x,x')=1}} \mathcal{L}(f_\theta(x'), y; \theta)\right] \quad (5)$$

In other words, the regularizer above aims to measure the expected value (across the training data) for the worst-case loss of fairness due to points x' that are assigned to different classes. Indeed, the regularizer (5) is not differentiable (with respect to the weights θ) due to the existence of the max operator. Nevertheless, one can compute an upper bound of (5) and aims to minimize this upper bound instead. Such upper bound can be derived as follows:

$$\begin{aligned}\max_{\substack{x'\in D_\phi\\ d(x,x')=1}} \mathcal{L}(f_\theta(x'), y; \theta) &= \max_{\substack{x'\in D_\phi\\ d(x,x')=1}} \begin{cases} -\log(1-f_\theta(x')) & \text{if } y=0\\ -\log(f_\theta(x')) & \text{if } y=1\end{cases}\\ &\le \begin{cases} -\log(1-\theta^T\overline{w}_{S_\phi(x)}) & \text{if } y=0\\ -\log(\theta^T\underline{w}_{S_\phi(x)}) & \text{if } y=1\end{cases}\end{aligned} \quad (6)$$

where $\theta^T\overline{w}_{S_\phi(x)}$ and $\theta^T\underline{w}_{S_\phi}$ are the linear upper/lower bound of $f_\theta(x')$ inside the set $S_\phi(x) = \{x' \in D_\phi | d(x,x') = 1\}$. Such linear upper/lower bound of $f_\theta(x')$ can be computed using off-the-shelf bounding techniques like Symbolic Interval Analysis (Wang et al. 2018a) and α-Crown (Xu et al. 2020). We denote by $\overline{\mathcal{L}}(y;\theta)$ the right hand side of the inequality in (6) which depends only on the label y and the NN parameters θ. Now the fairness property can be incorporated into training by optimizing the following problem over θ (the NN parameters):

$$\min_\theta \mathbb{E}_{(x,y)\in(X,Y)}\left[(1-\lambda_f)\underbrace{\mathcal{L}(f_\theta(x), y;\theta)}_{\text{natural loss}} + \lambda_f \underbrace{\overline{\mathcal{L}}(y;\theta)}_{\substack{\text{local}\\ \text{fairness loss}}}\right], \quad (7)$$

where λ_f is a regularization parameter to control the trade-off between fairness and accuracy.

Although easy to compute and incorporate in training, the regularizer $\mathcal{L}_f^l(\theta)$ (and its upper bound) defined above suffers from a significant drawback. It focuses on the fairness *around* the samples presented in the training data. In other words, although the aim was to promote *global* fairness, this regularizer is effectively penalizing the training only in the *local* neighborhood of the training data. Therefore, its effectiveness depends greatly on the quality of the training data and its distribution. Poor data distribution may lead to the poor effect of this regularizer. Next, we introduce another regularizer that avoids such problems.

Global Fairness Regularizer $\mathcal{L}_f^g$ using Product Network: To avoid the dependency on data, we introduce a novel fairness regularizer capable of capturing global fairness during the training. Such a regularizer is made possible thanks to the product NN defined above. In particular, the global fairness regularizer $\mathcal{L}_f^g(\theta)$ is defined as:

$$\mathcal{L}_f^g(\theta) = \max_{\substack{(x,x')\in D_\phi\times D_\phi\\ d(x,x')=1}} |f_\theta(x) - f_\theta(x')| \quad (8)$$

In other words, the regularizer $\mathcal{L}_f^g(\theta)$ in (8) aims to penalize the worst case loss in global fairness. Similar to (5), the $\mathcal{L}_f^g(\theta)$ is also non-differentiable with respect to θ. Nevertheless, thanks to the product NN, it can be bounded as:

$$\begin{aligned}\mathcal{L}_f^g(\theta) &\le \max_{(x,x')\in D_\phi\times D_\phi} |f_\theta(x) - f_\theta(x')|\\ &= \max_{x_p\in D_\phi\times D_\phi} f_p(x_p) \le \theta^T\overline{w}_{D_\phi}\end{aligned} \quad (9)$$

where $\theta^T\overline{w}_{D_\phi}$ is the linear upper bound of the product network among the domain $D_\phi \times D_\phi$. Again, such bound can be computed using Symbolic Interval Analysis or α-Crown on the product network after replacing the output h_p with $f_p = |f_\theta(x) - f_\theta(x')|$. It is crucial to note that the upper bound in (9) depends only on the domain D_ϕ. Hence, the fairness property can now be incorporated into training by minimizing this upper bound as:

$$\min_\theta \mathop{\mathbb{E}}_{(x,y)\in(X,Y)}\left[(1-\lambda_f)\underbrace{\mathcal{L}(f_\theta(x), y;\theta)}_{\substack{\text{natural}\\ \text{loss}}}\right] + \lambda_f \underbrace{\theta^T\overline{w}_{D_\phi}}_{\substack{\text{global}\\ \text{fairness loss}}}, \quad (10)$$

where the fairness loss is now outside the $\mathbb{E}[\ .\ ]$ operator.

In the next section, we show that the global fairness regularizer $\mathcal{L}_f^g(\theta)$ empirically outperforms the local fairness regularizer $\mathcal{L}_f^l(\theta)$. We end up our discussion in this section with the following result:

Proposition 4.2. *Consider a NN model f_θ and a fairness property ϕ—either representing a Global Individual Fairness property (Definition 2.1) or an Individual Discrimination property (Definition 2.2). Consider a NN model f_θ trained using the objective function in* (10). *If $\theta^T\overline{w}_{D_\phi} = 0$ by the end of the training, then the resulting f_θ is guaranteed to satisfy ϕ.*

Proof. The result follows directly from equation (9). □

Indeed, the result above is just a sufficient condition. In other words, the NN may still satisfy the fairness property ϕ even if $\theta^T\overline{w}_{D_\phi} > 0$. Such cases can be handled by applying the verification procedure in Section 4.2 after training.

5 Experimental Evaluation

We present an experimental evaluation to study the effect of our proposed fairness regularizers and hyperparameters on global fairness. We evaluated CertiFair on four widely investigated fairness datasets (Adult (Dua and Graff 2017), German (Dua and Graff 2017), Compas (Angwin et al. 2016), and Law School (Wightman 1998)). All datasets were pre-processed such that any missing rows or columns were dropped, features were scaled so that they are between $[0, 1]$, and categorical features were one-hot encoded.

Implementation: We implemented our framework in a Python tool called CertiFair that can be used for training and verification of NNs against an individual fairness property. CertiFair uses Pytorch (Paszke et al. 2019) for NN training and PeregriNN as a NN verifier. We run all our experiments

δ_i	0.03	0.05	0.07	0.1
Certified Local Fairness	100.00	81.42	100.00	99.95
Certified Global Fairness	65.98	6.40	57.39	66.35

Table 1: Comparison between local and global fairness on the Adult dataset for different networks and similarity constraints (distance δ_i). The training was done using the local fairness regularizer.

using a single GeForce RTX 2080 Ti GPU and two 24-core Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz (only 8 cores were used for these experiments).

Measuring global fairness using verification: While the verifier is capable of finding concrete counterexamples that violate the fairness, it is also important to *quantify* a bound on the fairness. In these experiments, the certified global fairness is quantified as the percentage of partitions of the input space with zero counterexamples. In particular, the input space is partitioned using the categorical features, i.e., the number of partitions is equal to the number of different categorical assignments and each partition corresponds to one categorical assignment. Note that the numerical features don't have to be fixed inside each partition (property dependent). To verify the property globally, we run the verifier on each partition of the input space and verify the fairness property. Finally, we count the number of verified fair partitions (normalized by the total number of partitions). We would like to note that partitioning the input space can be a bottleneck for verification for large domains due to the exponential number of partitions. Our experiments (Appendix D[2]) show that for the most commonly used datasets, our framework is able to train and certify networks in a very reasonable time.

Fairness properties: In the experimental evaluation, we consider two classes of fairness properties. The first class $\mathcal{P}_1$ is the one in definition 2.2 where two individuals are similar if they only differ in their sensitive attribute. The second class of properties $\mathcal{P}_2$ relaxes the first by also allowing numerical attributes to be close (not identical), this is allowed under definition 2.1 of global individual fairness by setting $\delta_i > 0$ for numerical attributes. Complete formal definitions of all the properties for the four datasets (full description in Appendix B[2]) is provided in Appendix C[2].

5.1 Experiment 1: Global Individual Fairness vs. Local Individual Fairness

In this experiment, we empirically show that NNs with high *local* individual fairness do not necessarily result in NNs with *global* individual fairness. In particular, we train several different NNs on the Adult dataset and considered multiple fairness properties (all from class $\mathcal{P}_2$ defined above) by varying δ_i. Note that δ_i is equal for all features i within the same property, but is different from one property to another. Next, we use PeregeriNN verifier to find counterexamples for both the *local* fairness (by applying the verifier to the trained NN) and the *global* fairness (by applying the verifier to the product NN). We measure the fairness of the NN for both cases and report the results in Table 1. The results indicate that verifying *local* fairness may result in incorrect conclusions about the fairness of the model. In particular, rows 2 and 4 in the table show that counterexamples were not found in the neighborhood of the training data (reflected by the 100% certified local fairness), yet verifying the product NN was capable of finding counterexamples that are far from the training data leading to accurate conclusions about the NN fairness.

5.2 Experiment 2: Effects of Incorporating the Fairness Regularizer

We investigate the effect of using the global fairness regularizer (defined in (9)) on the decisions of the NN classifier when trained on the Adult dataset. The fairness property for this experiment is of class $\mathcal{P}_1$. To investigate the predictor's bias, we first project the data points on two numerical features (age and hours/week). Our objective is to check whether the points that are classified positively for the privileged group are also classified positively for the non-privileged group. Figure 2 (left) shows the predictions for the unprivileged group when using the base classifier ($\lambda_f = 0$). Green markers indicate points for which individuals from both privileged and non-privileged groups are treated equally. The red markers show individuals from the non-privileged group that did not receive the same NN output compared to the corresponding identical ones in the privileged group. Figure 2 (right) shows the same predictions but using the fair classifier ($\lambda_f = 0.03$), the predictions from this classifier drastically decreased the discrimination between the two groups while only decreasing the accuracy by 2%. These results suggest that we can indeed regularize the training to improve the satisfaction of the fairness constraint without a drastic change in performance.

We also investigate how the certified fairness changes across epochs of training. To that end, we train a NN for the Adult dataset and evaluate the test accuracy as well as the certified global fairness after each epoch of training for two different values of λ_f. Figure 3 shows the underlying trade-off between achieving fairness versus maximizing accuracy. As expected, lower values of λ_f result in a lower loss in accuracy while having a lower effect on fairness. The results also show that a small sacrifice of the accuracy can lead to a significant enhancement of the fairness as shown for the $\lambda_f = 0.007$ case.

5.3 Experiment 3: Certified Fair Training

Experiment setup: The objective of this experiment is to compare the two regularizers, the local fairness regularizer in (6) and the global fairness regularizer in (9). To that end, we performed a grid search over the learning rate α, the fairness regularization parameter λ_f, and the NN architecture to get the best test accuracy across all datasets. The best performance was obtained with a NN that consists of two hidden layers of 20 neurons (except for the German dataset, where we use 30 neurons per layer), learning rate $\alpha = 0.001$, global fairness regularization parameter λ_f equal to 0.01 for Adult and Law School, 0.005 for German, and 0.1 for

[2]Extended version at https://arxiv.org/abs/2205.09927

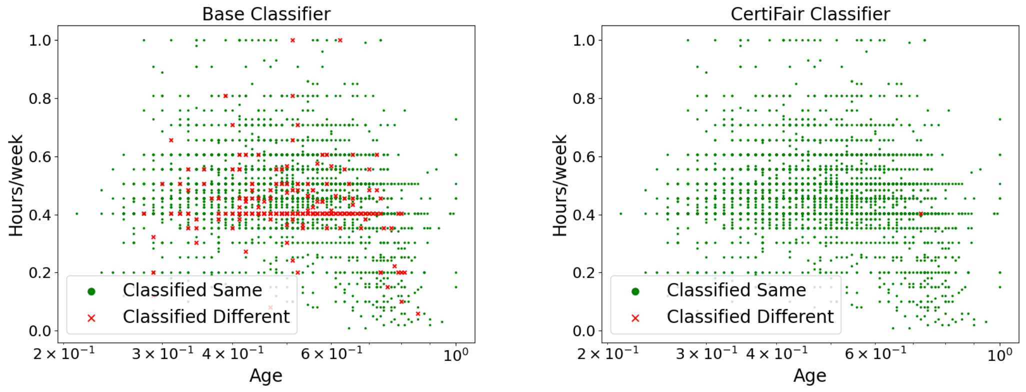

Figure 2: Comparison between the base and CertiFair classifiers in terms of fairness as defined in 2.2. We show the classifications for the minority group of the adult dataset projected on two features; Age and hours worked per week. The figure shows that the base classifier suffers from biases against identical individuals who are of a different race (red markers). CertiFair is able to drastically improve the individual fairness on this dataset with only 2% reduction in accuracy.

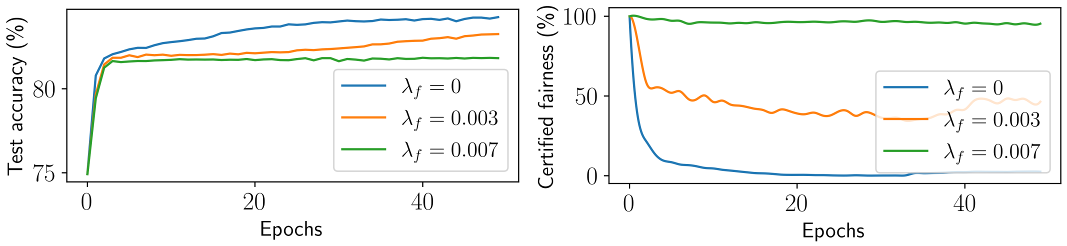

Figure 3: Test accuracy (left) and certified fairness (right) across training epochs when training a NN with the fairness constraint ($\lambda_f = 0.003$, $\lambda_f = 0.007$) and without it ($\lambda_f = 0$) on the Adult dataset.

Compas dataset, and local fairness regularization parameter λ_f equal to 0.95 for Adult, 0.9 for Compas, 0.2 for German, and 0.5 for Law School. We trained the models for 50 epochs with a batch size of 256 (except for Law School, where the batch size is set to 1024) and used Adam Optimizer for learning the NN parameters θ. All the datasets were split into 70% and 30% for training and testing. Additional experiments (Appendix D.2[2]) compares the proposed regularizers with LCIFR.

Effect of the choice of the fairness regularizer: We investigate the certified global fairness for the two regularizers introduced in Section 4.3. Table 2 summarizes the results for $\mathcal{P}_1$ and $\mathcal{P}_2$ fairness properties (defined in details in Appendix C[2]) across different datasets. For each property and dataset, we compare the test accuracy, positivity rate (percentage of points classified as 1), and the certified global fairness of the base classifier (trained with $\lambda_f = 0$) and the CertiFair classifier trained twice with two different fairness regularizers $\mathcal{L}_f^l$ and $\mathcal{L}_f^g$. Compared to the base classifier, training the NN with the global fairness regularizer $\mathcal{L}_f^g$ significantly increases the certified global fairness with a small drop in the accuracy in most of the cases except for the Law School dataset, where the test accuracy dropped by 7 % on $\mathcal{P}_2$ but the global fairness increased by 45 %. Compared to the local regularizer $\mathcal{L}_f^l$, the global regularizer achieves higher global fairness and comparable (if not better) test accuracy on all datasets except Law School. We think that this might be due to the network's limited capacity to optimize both objectives. We also report the positivity rate (number of data points classified positively) for the classifiers. This metric is important because most of these datasets are unbalanced, and hence the classifiers can trivially skew all the classifications to a single label and achieve high fairness percentage. Thus it is desired that the positivity rate of the CertiFair classifier to be close to the one of the base classifier to ensure that it is not trivial. Lastly, we conclude that even though the local regularizer improves the global fairness, the global regularizer can achieve higher degrees of certified global fairness without a significant decrease in test accuracy, and of course, it avoids the drawbacks of the local regularizer discussed in Section 4.3.

Effect of the fairness regularization parameter: In this experiment, we investigate the effect of the fairness regularization parameter λ_f on the classifier's accuracy and fairness. The parameter λ_f controls the trade-off between the accuracy of the classifier and its fairness, and tuning this parameter is usually dependent on the network/dataset. To that end, we trained a two-layer NN with 30 neurons per layer for the German dataset using 8 different values for λ_f and summarized the results in Table 3. The fairness property verified

Constraint	Dataset	Test Accuracy (%)			Positivity Rate(%)			Certified Global Fairness (%)		
		Base	$\mathcal{L}_f^g$	$\mathcal{L}_f^l$	Base	$\mathcal{L}_f^g$	$\mathcal{L}_f^l$	Base	$\mathcal{L}_f^g$	$\mathcal{L}_f^l$
$\mathcal{P}_1$	Adult	84.55	82.33	83.25	20.80	18.13	20.15	1.77	97.86	95.31
	German	75.30	72.66	69.66	79.00	83.14	81.71	14.81	92.59	82.71
	Compas	68.30	65.08	63.82	61.55	66.00	71.60	47.22	100.00	100.00
	Law	87.60	84.90	86.69	21.42	17.50	21.63	34.16	70.10	86.45
$\mathcal{P}_2$	Adult	84.55	82.34	83.81	20.8	17.4	21.79	6.40	100.00	61.92
	German	75.30	73.00	70.00	79.00	72.00	83.00	8.64	95.06	86.41
	Compas	68.30	65.08	63.19	61.55	66.00	69.40	11.14	100.00	100.00
	Law	87.60	79.92	78.69	21.51	9.10	25.03	6.87	51.87	78.54

Table 2: Comparison between a base classifier ($\lambda_f = 0$) and CertiFair classifier with different fairness regularizers.

λ_f	Test accuracy (%)	Certified global fairness (%)
1×10^{-4}	74.33	8.64
5×10^{-4}	74.33	14.81
5×10^{-3}	72.33	66.66
7×10^{-3}	72.66	82.71
1×10^{-2}	73.00	95.06
5×10^{-2}	66.33	100.00

Table 3: Effect of fairness regularization parameter λ_f on the test accuracy and certified fairness.

is of class $\mathcal{P}_2$. The results show that the global fairness satisfaction can increase without a significant drop in accuracy up to a certain point, after which the fairness loss is dominant and results in a significant decrease in the accuracy.

6 Related work

Group fairness: Group fairness considers notions like demographic parity (Feldman et al. 2015), equality of odds, and equality of opportunity (Hardt, Price, and Srebro 2016). Tools that verify notions of group fairness assume knowledge of a probabilistic model of the population. FairSquare (Albarghouthi et al. 2017) relies on numerical integration to formally verify notions of group fairness; however, it does not scale well for NNs. VeriFair (Bastani, Zhang, and Solar-Lezama 2019) considers probabilistic verification using sampling and provides soundness guarantees using concentration inequalities. This approach is scalable to big networks, but it does not provide worst-case proof.

Individual fairness: LCIFR (Ruoss et al. 2020) proposes a technique to learn fair representations that are provably certified. An encoder is *empirically* trained to map similar individuals to be within the neighborhood of the given individual and then apply NN verification techniques to this neighborhood to certify fairness. The property verified is a *local* property with respect to the given individual. On the contrary, our work focuses on the global fairness properties of a NN. It also avoids the empirical training of similarity maps to avoid affecting the soundness and completeness of the proposed framework. In the context of individual global fairness, a recent work (John, Vijaykeerthy, and Saha 2020) proposed a sound but incomplete verifier for linear and kernelized polynomial/radial basis function classifiers. It also proposed a meta-algorithm for the global individual fairness verification problem; however, it is not clear how it can be used to design sound and complete NN verifiers for the fairness properties. Another line of work (Urban et al. 2020) focuses on proving *dependency* fairness properties which is a more restrictive definition of fairness since it requires the NN outputs to avoid any dependence on the sensitive attributes. The method employs forward and backward static analysis and input space partitioning to verify the fairness property. This definition of fairness is different from the individual fairness we are considering in this work and is more restrictive. DIF (Yurochkin and Sun 2021) considers an average case fairness which is unlike our work which considers worst-case fairness guarantees. Lastly, a very recent work (Benussi et al. 2022) considers a similar definition of global individual fairness problem. However, the framework formulates the problem using MILP which is known to suffer from scalability issues. Moreover, it introduces a local regularizer that uses MILP for bounds computation. This regularizer, though powerful, is impractical due to scalability issues. For example, a single layer network with 100 neurons takes 9 hours to train. Unlike this work, Certifair introduces a global regularizer that is data independent and uses over-approximation techniques during verification which are known to be far more scalable than MILP-based methods. Appendix D.4^2 shows the execution time needed to train models that are orders of magnitude larger than those in (Benussi et al. 2022). The worst case is less than 20 minutes which is orders of magnitudes better than the 9 hours needed in (Benussi et al. 2022) albeit for much larger architectures.

NN verification: This work is algorithmically related to NN verification in the context of adversarial robustness. However, robustness is a local property of a NN given an input, and a norm bounded perturbation. Moreover, the robustness property does not consider the notion of sensitive attributes. The NN verification literature is extensive (Katz et al. 2019, 2017; Ehlers 2017; Lomuscio and Maganti 2017; Tjeng, Xiao, and Tedrake 2019; Bastani et al. 2016; Bunel et al. 2020; Fischetti and Jo 2018; Anderson et al. 2020; Cheng, Nührenberg, and Ruess 2017; Xiang, Tran, and Johnson 2017, 2018; Gehr et al. 2018; Wang et al. 2018b; Tran et al. 2020; Ivanov et al. 2019; Fazlyab et al. 2019; Wang et al. 2018a; Dvijotham et al. 2018; Wong and Kolter 2018; Henriksen and Lomuscio 2021; Wang et al. 2021; Singh et al. 2019) and any of those verifiers can be modified to verify fairness properties of the Product NN.

Acknowledgments

The authors acknowledge funding from NSF Awards #2002405 and #2139781.

References

Aggarwal, A.; Lohia, P.; Nagar, S.; Dey, K.; and Saha, D. 2019. Black Box Fairness Testing of Machine Learning Models. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2019, 625–635. New York, NY, USA: Association for Computing Machinery. ISBN 9781450355728.

Albarghouthi, A.; D'Antoni, L.; Drews, S.; and Nori, A. V. 2017. FairSquare: Probabilistic Verification of Program Fairness. *Proc. ACM Program. Lang.*, 1(OOPSLA).

Anderson, R.; Huchette, J.; Ma, W.; Tjandraatmadja, C.; and Vielma, J. P. 2020. Strong mixed-integer programming formulations for trained neural networks. *Mathematical Programming*, 183(1): 3–39.

Angwin, J.; Larson, J.; Mattu, S.; and Kirchner, L. 2016. How We Analyzed the COMPAS Recidivism Algorithm. https://www.propublica.org/article/how-we-analyzed-the-compas-recidivism-algorithm. Accessed: 2022-05-01.

Barthe, G.; Crespo, J. M.; and Kunz, C. 2011. Relational verification using product programs. In *International Symposium on Formal Methods*, 200–214. Springer.

Bastani, O.; Ioannou, Y.; Lampropoulos, L.; Vytiniotis, D.; Nori, A.; and Criminisi, A. 2016. Measuring Neural Net Robustness with Constraints. In *Advances in Neural Information Processing Systems*, volume 29, 2613–2621.

Bastani, O.; Zhang, X.; and Solar-Lezama, A. 2019. Probabilistic verification of fairness properties via concentration. *Proceedings of the ACM on Programming Languages*, 3(OOPSLA): 1–27.

Benussi, E.; Patane, A.; Wicker, M.; Laurenti, L.; and Kwiatkowska, M. 2022. Individual Fairness Guarantees for Neural Networks. *arXiv preprint arXiv:2205.05763*.

Brennan, T.; Dieterich, W.; and Ehret, B. 2009. Evaluating the predictive validity of the COMPAS risk and needs assessment system. *Criminal Justice and behavior*, 36(1): 21–40.

Bunel, R.; Lu, J.; Turkaslan, I.; Kohli, P.; Torr, P.; and Mudigonda, P. 2020. Branch and bound for piecewise linear neural network verification. *Journal of Machine Learning Research*, 21(42): 1–39.

Cheng, C.-H.; Nührenberg, G.; and Ruess, H. 2017. Maximum Resilience of Artificial Neural Networks. In D'Souza, D.; and Narayan Kumar, K., eds., *Automated Technology for Verification and Analysis*, 251–268. Springer.

Dua, D.; and Graff, C. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml. Accessed: 2022-05-01.

Dvijotham, K.; Stanforth, R.; Gowal, S.; Mann, T. A.; and Kohli, P. 2018. A Dual Approach to Scalable Verification of Deep Networks. In Globerson, A.; and Silva, R., eds., *Uncertainty in Artificial Intelligence*, volume 1, 550–559. ISBN 978-0-9966431-3-9.

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

Ehlers, R. 2017. Formal verification of piece-wise linear feed-forward neural networks. In D'Souza, D.; and Kumar, K. N., eds., *International Symposium on Automated Technology for Verification and Analysis*, 269–286. Springer.

Fazlyab, M.; Robey, A.; Hassani, H.; Morari, M.; and Pappas, G. 2019. Efficient and accurate estimation of lipschitz constants for deep neural networks. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 32, 11423–11434. Curran Associates, Inc.

Feldman, M.; Friedler, S. A.; Moeller, J.; Scheidegger, C.; and Venkatasubramanian, S. 2015. Certifying and removing disparate impact. In *proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 259–268.

Fischetti, M.; and Jo, J. 2018. Deep neural networks and mixed integer linear optimization. *Constraints*, 23(3): 296–309.

Gehr, T.; Mirman, M.; Drachsler-Cohen, D.; Tsankov, P.; Chaudhuri, S.; and Vechev, M. 2018. AI2: Safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE Symposium on Security and Privacy (SP)*, 3–18. IEEE.

Hardt, M.; Price, E.; and Srebro, N. 2016. Equality of opportunity in supervised learning. *Advances in neural information processing systems*, 29.

Henriksen, P.; and Lomuscio, A. 2021. DEEPSPLIT: An Efficient Splitting Method for Neural Network Verification via Indirect Effect Analysis. In *IJCAI*, 2549–2555.

Ivanov, R.; Weimer, J.; Alur, R.; Pappas, G. J.; and Lee, I. 2019. Verisig: verifying safety properties of hybrid systems with neural network controllers. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, HSCC '19, 169–178. New York, NY, USA: Association for Computing Machinery.

John, P. G.; Vijaykeerthy, D.; and Saha, D. 2020. Verifying individual fairness in machine learning models. In *Conference on Uncertainty in Artificial Intelligence*, 749–758. PMLR.

Katz, G.; Barrett, C.; Dill, D. L.; Julian, K.; and Kochenderfer, M. J. 2017. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. In Majumdar, R.; and Kunčak, V., eds., *Computer Aided Verification*, Lecture Notes in Computer Science, 97–117. Springer International Publishing. ISBN 978-3-319-63387-9.

Katz, G.; Huang, D. A.; Ibeling, D.; Julian, K.; Lazarus, C.; Lim, R.; Shah, P.; Thakoor, S.; Wu, H.; Zeljić, A.; et al. 2019. The marabou framework for verification and analysis of deep neural networks. In Dillig, I.; and Tasiran, S., eds., *Computer Aided Verification*, 443–452. Springer International Publishing.

Khedr, H.; Ferlez, J.; and Shoukry, Y. 2021. PEREGRiNN: Penalized-Relaxation Greedy Neural Network Verifier. In Silva, A.; and Leino, K. R. M., eds., *Computer Aided Verification*, 287–300. Cham: Springer International Publishing. ISBN 978-3-030-81685-8.

Liu, C.; Arnon, T.; Lazarus, C.; Strong, C.; Barrett, C.; Kochenderfer, M. J.; et al. 2021. Algorithms for verifying deep neural networks. *Foundations and Trends® in Optimization*, 4(3-4): 244–404.

Lomuscio, A.; and Maganti, L. 2017. An approach to reachability analysis for feed-forward ReLU neural networks. arXiv:1706.07351.

Mehrabi, N.; Gupta, U.; Morstatter, F.; Steeg, G. V.; and Galstyan, A. 2021a. Attributing Fair Decisions with Attention Interventions. *arXiv preprint arXiv:2109.03952*.

Mehrabi, N.; Morstatter, F.; Saxena, N.; Lerman, K.; and Galstyan, A. 2021b. A survey on bias and fairness in machine learning. *ACM Computing Surveys (CSUR)*, 54(6): 1–35.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Kopf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, 8024–8035. Curran Associates, Inc.

Ruoss, A.; Balunovic, M.; Fischer, M.; and Vechev, M. 2020. Learning certified individually fair representations. *Advances in Neural Information Processing Systems*, 33: 7584–7596.

Singh, G.; Gehr, T.; Püschel, M.; and Vechev, M. 2019. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages*, 3(POPL): 1–30.

Tjeng, V.; Xiao, K.; and Tedrake, R. 2019. Evaluating Robustness of Neural Networks with Mixed Integer Programming. arXiv:1711.07356.

Tran, H.-D.; Yang, X.; Manzanas Lopez, D.; Musau, P.; Nguyen, L. V.; Xiang, W.; Bak, S.; and Johnson, T. T. 2020. NNV: The Neural Network Verification Tool for Deep Neural Networks and Learning-Enabled Cyber-Physical Systems. In Lahiri, S. K.; and Wang, C., eds., *Computer Aided Verification*, 3–17. Springer International Publishing.

Urban, C.; Christakis, M.; Wüstholz, V.; and Zhang, F. 2020. Perfectly Parallel Fairness Certification of Neural Networks. *Proc. ACM Program. Lang.*, 4(OOPSLA).

Wang, S.; Pei, K.; Whitehouse, J.; Yang, J.; and Jana, S. 2018a. Efficient formal safety analysis of neural networks. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 31, 6367–6377.

Wang, S.; Pei, K.; Whitehouse, J.; Yang, J.; and Jana, S. 2018b. Formal security analysis of neural networks using symbolic intervals. In *Proceedings of the 27th USENIX Conference on Security Symposium*, SEC'18, 1599–1614. USENIX Association.

Wang, S.; Zhang, H.; Xu, K.; Lin, X.; Jana, S.; Hsieh, C.-J.; and Kolter, J. Z. 2021. Beta-crown: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification. *arXiv preprint arXiv:2103.06624*.

Wightman, L. F. 1998. *LSAC national longitudinal bar passage study*. Law School Admission Council.

Wong, E.; and Kolter, Z. 2018. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International conference on machine learning*, 5286–5295. PMLR.

Xiang, W.; Tran, H.-D.; and Johnson, T. T. 2017. Reachable Set Computation and Safety Verification for Neural Networks with ReLU Activations. arXiv:1712.08163.

Xiang, W.; Tran, H.-D.; and Johnson, T. T. 2018. Output reachable set estimation and verification for multilayer neural networks. *IEEE transactions on neural networks and learning systems*, 29(11): 5777–5783.

Xu, D.; Yuan, S.; Zhang, L.; and Wu, X. 2018. Fairgan: Fairness-aware generative adversarial networks. In *2018 IEEE International Conference on Big Data (Big Data)*, 570–575. IEEE.

Xu, K.; Zhang, H.; Wang, S.; Wang, Y.; Jana, S.; Lin, X.; and Hsieh, C.-J. 2020. Fast and complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers. *arXiv preprint arXiv:2011.13824*.

Yurochkin, M.; Bower, A.; and Sun, Y. 2020. Training individually fair ML models with sensitive subspace robustness. In *International Conference on Learning Representations*.

Yurochkin, M.; and Sun, Y. 2021. SenSeI: Sensitive Set Invariance for Enforcing Individual Fairness. In *International Conference on Learning Representations*.

Zhang, B. H.; Lemoine, B.; and Mitchell, M. 2018. Mitigating unwanted biases with adversarial learning. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, 335–340.

Zhang, H.; Chen, H.; Xiao, C.; Gowal, S.; Stanforth, R.; Li, B.; Boning, D.; and Hsieh, C.-J. 2019. Towards stable and efficient training of verifiably robust neural networks. *arXiv preprint arXiv:1906.06316*.