Provable Fairness for Neural Network Models using Formal Verification

Anonymous Authors

***Abstract*—Machine learning models are increasingly deployed for important decision-making tasks, making it important to verify that they do not contain gender or racial biases picked up from training data. Typical approaches to achieve fairness revolve around efforts to clean or curate training data, with post-hoc statistical evaluation of the fairness of the model on evaluation data. In contrast, we propose techniques to *prove* the fairness properties using recently developed formal methods that verify properties of neural network models. Beyond the strength of guarantee implied by a formal proof, our methods have the advantage that we do not need explicit training or evaluation data (which is often proprietary) in order to analyze a given trained model. In experiments on two familiar datasets in the fairness literature (COMPAS and ADULTS), we show that through proper training, we can reduce unfairness by an average of 65.4% at a cost of less than 1% in AUC score.**

Keywords: Neural Network Verification, Provable Fairness

I. Introduction

Machine learning models are increasingly deployed for important decision-making tasks, making it important to verify that they do not contain gender or racial biases picked up from training data. Typical approaches to achieve fairness revolve around efforts to clean or curate training data, with post-hoc statistical evaluation of the fairness of the model on evaluation data.

In contrast, we propose techniques to prove the fairness properties using recently developed formal methods that verify properties of neural network models. Beyond the strength of guarantee implied by a formal proof, our methods have the advantage that we do not need explicit training or evaluation data (which is often proprietary) to analyze a given trained model.

Formal approaches to neural network verification have been developed over the past few years [10], [12], [13], [20] that can in certain cases prove properties about all possible executions of a specific network. Existing work has focused on networks used in safety-critical control algorithms and robotics [11], [25], or guaranteed robustness for visual perception networks [19].

In this paper, we build upon such methods to the task of validating fairness properties of classification and regression models. We systematically explore classes of fairness specifications that can be evaluated both statistically as well as using the developed formal approaches.

We consider using formal methods to analyze fairness for *classification tasks with explicit input labels*. Here one of the input fields for each record explicitly encodes a sensitive property, such as race, gender, or age. Informally, fairness dictates that a given model M should perform "the same" for all possible values of these sensitive fields. Note that excluding such inputs when training a model does not solve the fairness problem, as sensitive properties are often inferable from other input variables (e.g. the zip code of an African-American neighborhood).

Formal analysis approaches are most often applied to fully connected networks with ReLU activation functions [2]. For such networks, a binary decision model partitions the space of possible inputs into a collection of geometric polytopes, such that all points within a particular polytope are labeled homogeneously as all positive or all negative. The fairness properties which we are concerned with revolve around showing that these geometric regions are comparable for two distinct labels or groups, as illustrated in Figure 1. There are several possible provable properties depending upon how we define that M performs "the same" for different groups, including:

- *Volumetric identity* – Here we say that M is provably fair when the positive regions are identical for different groups A and B, ignoring the input variable explicitly encoding the identity of group A and B.
- *Volumetric magnitude* — Here we say M is provably fair when the volume of the positive regions are comparable for different groups A and B. The symmetric difference of the positive regions of A and B may be greater than zero, but the difference of the size of the input volumes is provably bounded.
- *Probability-weighted magnitude* — Here we say that M is provably fair with respect to input distributions A and B when the weighted volumes of the positive regions are the same for A and B, or that the difference of these weighted volumes is provably bounded. The weighted volume of polytope P is defined by the integral of the probability of each point in P.
- *Probability-weighted symmetric difference* — Here we say that M is provably fair with respect to input

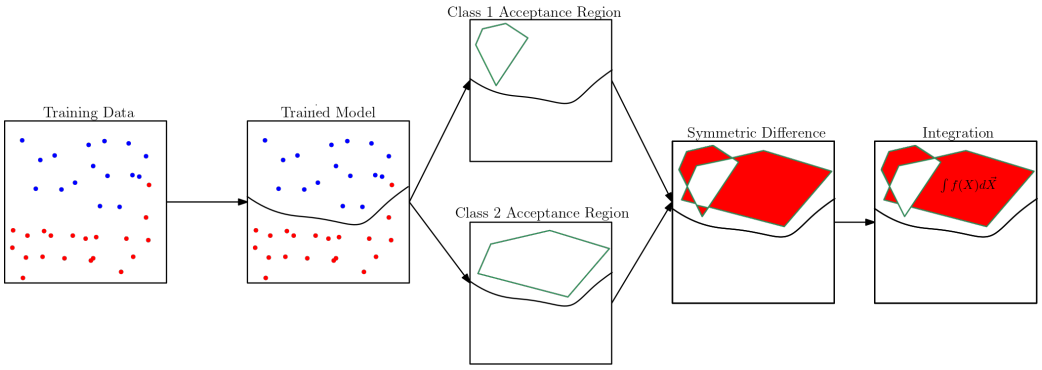

Fig. 1. Each partition Θ of the network input space is used as the domain for integration over a given input probability density function P, calculating a probability of the input space with a fixed output classification. Repeating and summing over all partitions, we can evaluate fairness metrics for the network.

distributions A and B if the percentage of each population which would change classified label by changing label of the protected class is below a certain threshold.

- *Volumetric symmetric difference* — Here, we say that M is provably fair if the volume of the non-overlapping acceptance regions for the protected classes is below a certain threshold.
- *Probability-weighted symmetric difference* — Here we say that M is provably fair with respect to input distributions A and B if the percentage of each population which would change classified label by changing label of the protected class is below a certain threshold.
- *Net Preference* — Here we say that M is provably fair with respect to input distributions A and B if there is not a large difference in acceptance probability if the criteria used to accept the population from distribution A is used evaluated on the population from distribution B.

Our major contributions in this paper are:

- *Provable Fairness Guarantees without Training or Evaluation Data* – We define a class of fairness guarantees which can be formally computed on neural classification models for real tasks.
  A particularly exciting aspect of our approach is that verification requires only black box access to the trained model. Previous approaches to fairness evaluation require access to training or evaluation data, usually deemed confidential and proprietary for consumer product models.
- *Retraining Techniques to Improve the Fairness of Models* – The fairness definitions we propose here can be embedded into the training process to help create fair machine learning systems, similar how methods for formal verification of perception systems have been used to create more robust vision systems [26].
  We measure the tradeoff between accuracy (as measured using AUC) and fairness (as measured using our proposed metrics) using four machine learning methods with varying levels of fairness-awareness. For two different tasks, we demonstrate that substantial increases of fairness can be achieved with little or no loss in accuracy. On the COMPAS dataset, our efforts to improve fairness reduced the average model quality (AUC score) by only 0.074% while reducing our three measures of unfairness (WSD, VSD, and NP, defined below) by an average of 66.1%, 70.3%, and 68.7%, respectively. On the ADULTS data set, reducing AUC by 0.44% improves fairness by 69.7%, 49.14%, and 68.27% respectively.
- *Addressing Scaling Issues* – Verifying the properties of neural networks requires sophisticated geometric algorithms in high-dimensional spaces. The number and complexity of polytopes defining our analysis grow exponential with the size of the model, and its parameter space.
  Still, the state-of-the-art in formal verification has advanced steadily despite theoretical intractability [12]. We identify the bottleneck in our approach as the volume computation step and propose alternate algorithms for volume integration that could further push the scalability limits of the approach.

This paper is organized as follows. Section II reviews previous work in formal verification techniques for neural networks, and also in the field of algorithmic fairness. We discuss our formal verification methods in Section III. Our experimental results demonstrating the trade off between model quality and quantified fairness are presented in Section IV. Finally, we do a critical performanace analysis on the scalability of our methods in Section IV-D, pointing to future research directions in the verification of model fairness. Our source code and datasets will be released upon publication, and included in this submission at https://bit.ly/3MC64Ke

II. Prior Work

Formal verification methods for neural networks perform set-based analysis of the network, rather than executing a network for individual input samples. Given a neural network $f_{NN} : \mathbb{R}^n \rightarrow \mathbb{R}^m$, an input set $X \subseteq \mathbb{R}^n$, and an unsafe set of outputs $U \subseteq \mathbb{R}^m$, the *open-loop NN verification problem* is to prove that for all allowed inputs $x \in X$, the network output is never in the unsafe set, $f_{NN}(x) \notin U$. One way to solve the verification problem is to compute the range of the neural network function for a specific input domain X, and then checking if the range intersects the unsafe states U [3], [22]. In this work, we reuse this range computation step in order to evaluate how probability distributions propagate through the networks.

Many algorithms [1], [13], [24] have been developed to perform neural network verification, including an annual competition to evaluate method applicability and scalability [2].

A related recent direction uses neural networks to estimate probability densities of reachable sets for motion planning [15], [16]. Here, the same two core operations were performed as in our work: (1) computing the range of the neural network as a union of polytopes, and (2) computing the volume of the output polytopes. This method, however, targeted probabilistic safety and motion planning, not fairness of classifiers.

Fairness in machine learning has also received increasing attention recently [5], [6], [14]. Existing work on fairness typically uses the input data and samples the network to provide estimates of fairness [4]. In contrast, our techniques use external distributions over input data and use set-based range computation of neural networks. Some proposed methods to improve fairness include using an adversary during training [23]. In this work, we explore data augmentation and selection strategies to improve fairness.

A limited amount of existing research also exists on provable fairness. One framework focuses on proving *dependency fairness* [9], [21], which strives to prove the outputs are not affected by certain input features. This method is based on forward and a backward static analysis as well as input feature partitioning. Another recent approach [17] focuses on *individual fairness*, which essentially means that similar individuals get similar treatments. In contrast, our fairness metrics are defined over geometric properties of the network outputs, such as the symmetric difference between the ranges with different values of sensitive inputs.

III. Methods

A. Acceptance Region

We evaluate the fairness of the models by looking at the acceptance region of the models for the classes of region. Let $f(\vec{x})$ be the label assigned by the neural network for the value $\vec{x}$ in the input space. The acceptance region for class C, which we denote as $\rho(C)$ is the region in the input space for which $\vec{x} \in C \wedge f(\vec{x}) = \ell$. The label ℓ that defines the acceptance region is user-definable based on the properties of the neural network they are interested in exploring. We denote the acceptance region of a class C as $\rho(C)$.

B. Metric

We next introduce the metrics that we use to evaluate the fairness of our models. The weighted symmetric difference (WSD) quantifies the total difference in the model's behaviour towards different classes. It is weighted in order to discount differences in behaviour for unrealistic sets of features that are improbable to ever appear in the input. Formally, the metric is defined as:

$$\begin{aligned}\mathrm{WSD}(C_1, C_2) &= \mathrm{Advantage}(C_1, C_2) \\ &\quad + \mathrm{Advantage}(C_2, C_1) \\ \mathrm{Advantage}(C_1, C_2) &= \int_{\rho(C_1)} P(\vec{X}|C_1) d\vec{X} \\ &\quad - \int_{\rho(C_1)\cap\rho(C_2)} P(\vec{X}|C_2) d\vec{X}\end{aligned}$$

The $\mathrm{Advantage}(C_1, C_2)$ metric quantifies how many examples from class C_1 would have not been in the acceptance region of the model if they were of class C_2 with all other features held equal.

The volumetric symmetric difference (VSD) also quantifies the total difference in the model's behaviour towards different classes. It may be used when information about the input feature distribution for the two classes is unavailable.

$$\begin{aligned}\mathrm{VSD}(C_1, C_2) &= |\rho(C_1) - \rho(C_2)| \\ &\quad + |\rho(C_2) - \rho(C_1)| \\ |\rho(C_1) - \rho(C_2)| &= \int_{\rho(C_1)} 1 \cdot d\vec{X} \\ &\quad - \int_{\rho(C_1)\cap\rho(C_2)} 1 \cdot d\vec{X}\end{aligned}$$

The net preference (NP) quantifies how much the model prefers the features of one class over the other when assigning a label. It is useful for investigating whether the model is making its decisions on variables strongly correlated with the protected class. When evaluating the preference, the acceptance region of C_1 is used in order to discount possible differences in the acceptance regions, which can be investigated using the

WSD and the VSD.

$$\begin{aligned}\text{NP}(C_1, C_2) &= \max\{|\,\text{Preference}(C_1, C_2)|,\\ &\quad |\,\text{Preference}(C_2, C_1)|\}\\ \text{Preference}(C_1, C_2) &= \int_{\rho(C_1)} P(\vec{X}|C_1)d\vec{X}\\ &\quad - \int_{\rho(C_1)} P(\vec{X}|C_2)d\vec{X}\end{aligned}$$

The preference function quantifies the difference in the acceptance probability of the two classes when evaluated by the acceptance critera of the first class.

C. Verification Approach

Our technical approach to verification is based on set-based execution of neural networks, extended to include probability distributions. Given a set of possible inputs, we propagate the set through the network to see the range of the network, the possible outputs. In our proposed approach, we will use the linear star set representation [7], [18] to propagate sets of states through the network. After analysis, the entire input set is partitioned into a number of polytopes that map to outputs with an identical label.

This representation of sets of inputs allows for defining a simple procedure for verifying the fairness of a neural network. There is an assumption that it is possible to encode C_1 and C_2 directly into the input features by fixing one or more of the features in the input to the appropriate values. The procedure is as follows:

1) Use neural-network reachability analysis in order to calculate the acceptance regions for each of the classes: $\rho(C1)$ and $\rho(C2)$.
2) Calculate $\rho(C_1) \cap \rho(C_2)$.
3) Calculate $\text{WSD}(C_1, C_2)$ using the probability distributions $P(\vec{X}|C1)$ and $P(\vec{X}|C_2)$ over the data.

The calculation of $\text{WSD}(C_1, C_2)$ requires a probability distribution over the data. For our experiments, we estimate the probability distribution from the input data, although external data could be used to create these distributions if input data is not available. The details of this method are in the appendix.

The neural network reachability analysis returns two lists of polytopes, each encoding the acceptance region of each class: $\rho(C_1) = \{A_1, A_2, A_3, \ldots A_N\}$ and $\rho(C_2) = \{B_1, B_2, B_3, \ldots B_M\}$. Calculating the intersection between the two regions is thus a matter of calculating the non-empty intersection between all pairs of polytopes in set induced by the Cartesian product $\rho(C_1) \times \rho(C_2)$. As the polytopes are $\mathcal{H}$-polytopes, they are specified as a list of constraints $C\alpha \leq d$. The constraints of two polytopes are simply concatenated to create the intersection of the two polytopes. Linear Programming can be used to determine if the intersection is empty.

D. Integration over the Probability Distribution

The final step for calculating the $\text{WSD}(C_1, C_2)$ for a model requires the integration the probability distributions $P(\vec{X}|C_1)$ and $P(\vec{X}|C_2)$ over a union of polytopes. In our experiments, direct integration of the probability distribution over the polytope proved too slow once more then two dimensions needed to be integrated over. However, the determining the volume of the polytope has a simpler time complexity then integrating over the volume of the polytope. Algorithms for doing so that are linear in the number of vertices exist, although we use QHull to find the volume due to its accessibility. We take advantage of this fact, computing the probability by discretizing the input space into a evenly-spaced grid. We calculate the intersection of each polytope with each region of the grid. We find the volume of this intersection and multiply it by the probability density at that point in the grid. This gives us a large speed-up in runtime for only a small cost in precision.

IV. Experimental Results

To investigate the fairness of the neural networks, we train various models on three datasets that are commonly used in model fairness studies, and measure their fairness with our method according to the metrics in Section III-B. Note that our measurement does not require expensive data sampling or availability of testing data, making it different compared to other works as described in Section II. Moreover, we also study two fairness-oriented data-augmentation strategies, effectively producing more *fair* models at little expense of predictive power.

In the following subsections, we present the statistics of the datasets and models, report both the performance and the fairness of the models trained by various data-augmentation strategies, including non-/fairness-enhanced.

A. Data Sets and Challenges

We selected three common datasets that contains both categorical and continuous variables as the input features. COMPAS [1] dataset records the risk score of defendants based on his/her demographics and prior criminal history. We train the models to predict whether the individual's risk score is above or below 5 as a binary classification task. HEALTH dataset contains the Charlson Index [2] based on the patient's age, sex and

[1]Correctional Offender Management Profiling for Alternative Sanctions

[2]Charlson Comorbidity Index predicts the ten-year mortality for a patient who may have a range of comorbid conditions

prior medical history. The models are trained to predict whether the Charlson Index is above or below the median value. ADULTS dataset contains the yearly income range (above or below $50K) of the individuals together with demographics, education and occupational information.

All the datasets contains at least one protected feature (e.g., Race or Sex) which we consider as a potential factor that incurs unfair inductive bias during model training. We could observe such unfairness in Table II and Table II where the WSD, VSD, and NP are non-zero. In the following subsection, we describe four training strategies to alleviate such unfairness.

B. Model Training

Four training strategies described as following:

1. *Original*: Use the original data to train the model without any augmentation or filtering.
2. *Protected Feature Permuted*: Randomly shuffle the protected feature (Race or Sex) in training data, then train the model once. The approach breaks the correlation between the feature and the label, for example, making the income prediction color blind.
3. *Data-Removed*: Re-train the model with the **removed** data, iteratively deleting the training data which may cross the decision boundary (threshold 0.5) when the protected feature is flipped.
4. *Data-Augmented*: Re-train the model with the **augmented** data, iteratively creating training data which may cross the decision boundary (threshold 0.5) when the protected feature is flipped. The max number of training data is limited to 3 times of the original data size.

The details of Strategy 3 and 4 can be found in Algorithm 1.

Algorithm 1 FAIRNESS ENHANCED DATA AUGMENTATION

Input: $s \in \{\text{Augment}, \text{Remove}\}$, $X^{(i)}_{train} \in \mathbb{R}^{n^{(i)} \times d}$, $\boldsymbol{y}^{(i)}_{train} \in \{0,1\}^{n^{(i)}}$, $i = 0$
while $i \leq 20$ and $1 \leq n^{(i)} \leq 3n^{(0)}$ **do**
  //Train model with current training data
  //$n^{(i)}$ is the #training data in i^{th} round
  $M^{(i)} = \text{TRAIN}(X^{(i)}_{train}, \boldsymbol{y}^{(i)}_{train})$
  //Based on current trained model, process data for the next round.
  $X^{(i+1)}_{train}, \boldsymbol{y}^{(i+1)}_{train} = \text{PROC}(X^{(i)}_{train}, \boldsymbol{y}^{(i)}_{train}, M^{(i)}, s)$
  i += 1
return $M^{(i)}$

procedure PROC($X, \boldsymbol{y}, M, s$)
  Input: $s \in \{\text{Augment}, \text{Remove}\}$, $X \in \mathbb{R}^{k \times d}$, $\boldsymbol{y} \in \{0,1\}^k$, M is the model
  $Ind_{flip} = []$
  for $i \in [k]$ **do**
    $y_i = M(\boldsymbol{x}_i)$
    //x'_i is the feature vector after flipping the protected feature.
    $y'_i = M(\boldsymbol{x}'_i)$
    if $y'_i \neq y_i$ **then**
      Ind_{flip}.append(i)
  if $s ==$ Remove **then**
    //Remove the data with different results after the feature flipped
    $X_{new} = \{\boldsymbol{x}_j | j \in [k], j \notin Ind_{flip}\}$
    $\boldsymbol{y}_{new} = \{y_j | j \in [k], j \notin Ind_{flip}\}$
    return $X_{new}, \boldsymbol{y}_{new}$
  if $s ==$ Augment **then**
    //Augment the data with the feature flipped and consistent label
    $X_{new} = \{\boldsymbol{x}'_j | j \in Ind_{flip}\}$
    $\boldsymbol{y}_{new} = \{y_j | j \in Ind_{flip}\}$
    $X_{new} = [X_{new}; X]$
    $\boldsymbol{y}_{new} = [\boldsymbol{y}_{new}; \boldsymbol{y}]$
    return $X_{new}, \boldsymbol{y}_{new}$

C. Results

There are several key results. The first is that without any measures to ensure fairness during the training process, the trained models demonstrate a significant degree of unfairness. Inspecting the WSD of various models trained on the COMPAS dataset in Table II, there is a significant number of individuals for whom the model decision would have switched had only their race been different. This demonstrates the importance of taking measures to ensure model fairness, especially in domains that have historically been race-sensitive.

The WSD and NP of various models on the ADULTS dataset in Table II demonstrate another important result. In the presence of many features on which to base its decision, a model might have small WSD, indicating that there are few individuals for which had their race been different, the model decision would have been different. However, a well-known problem is the ability to use a *proxy variable*, such as ZIP code and other correlated variables, in order to determine the race of an individual. This can be seen in the fact that the WSD is small but the NP is large for the original models in Table II, indicating that the model has a strong preference for features correlated with a certain race.

Comparisons of the WSD and the VSD show that when the probability distribution on the input features is not available, the VSD is a suitable proxy for the WSD. Reductions in the WSD are accompanied by similar reductions in the VSD. In 11 out of the 16 trials, the model with the smallest WSD was also the model with the smallest VSD.

Finally, our results demonstrate the effectiveness of

TABLE I
THE STATISTICS OF DATASET. FOR EACH DATASET WE SPLIT DATA INTO TRAIN/VAL/TEST SET WITH THE RATIO 70%/15%/15%, RESPECTIVELY. FOR EACH CONTINUOUS FEATURE, WE APPLIED MIN-MAX NORMALIZATION. FOR SIMPLIFICATION, WE USE BINARY LABEL FOR THE PROTECTED FEATURE: RACE (WHITE V.S. AFRICAN AMERICAN) AND SEX (MALE V.S. FEMALE).

Dataset	**COMPAS**	**ADULTS**	**HEALTH**
#Samples	2,363	43,031	124,086
#Categorical Feat.	2	5	11 (10 for one-hot)
#Continuous Feat.	2	3	11
Protected Feat.	Race	Race	Sex
#Classes	Binary	Binary	Binary

our data-augmentation method in order to improve fairness. The data-augmentation method leads to a significant improvement in the fairness of the models, with only a small drop in model accuracy (and sometimes an improvement). The data-augmentation method is often able to significantly outperform or achieve parity with the baseline of permuting the race in many cases for the WSD metric. However, unlike the race-permutation method, the method is also effective in greatly reducing the NP. Thus, the models rely much less on proxy variables for race.

D. Limits of Scalability

The health dataset was omitted due to scalability issues of the current method. Performing the reachability analysis and identifiying the union of polytopes corresponding to the acceptance region of each of the classes took a total of less than two minutes ($\sim$ 100s) for the 197 models tested for the health dataset. However, attempts to integrate the probability distribution over the probability distribution in a reasonable time frame failed. The integration method requires discretizing the input space, dividing each continuous dimension into x divisions. If there are d continuous dimensions, this creates x^d total divisions, which are iterated over. As there are 11 continuous dimensions in the health dataset, this generates x^{11} total divisions. The precision of the integration is determined by the number of divisions. For a reasonably precise answer, at least 10 divisions are necessary. With at least 10^{11} divisions per polytope, this naive integration method was impractical.

Another approximation for the integral of the probability distribution over the polytope was to sample the probability distribution at the center of the polytope and scaling it by the volume of the polytope. Our method for finding the volume requires enumerating the polytope vertices. This becomes infeasible for 11-dimensional polytopes as well.

There are alternative methods to finding the volume of the polytope that do not require enumeration of the vertices. Emeris and Fisikopoulos [8] give a polytope volume approximation algorithm that takes polynomial time in the desired precision. As the number of vertices of a polytope can grow exponentially with the dimension, this improves the practicality of our method.

E. Limits of Current Method

1) Integration: The procedure of discretizing the input space into a grid makes analysis infeasible for more than a small number of continuous inputs. If there are 11 continuous variables, and each axis is broken into 10 bins, this yields $10^1 1$ total regions. Thus, the number of continuous variables must be limited. There are workarounds to this approach. A direct discretization of the Polytope based on the gradient of the $P(\vec{X}|C)$ would greatly reduce the number of regions while even allowing for improvements in precision. Additionally, efficient methods for integration of *polynomials* over Polytopes exist, which can be exploited if the $P(\vec{X}|C)$ takes the form of a polynomial.

2) Handling of One-Hot Features: The manner in which one-hot features are handled in the reachability analysis is by analyzing the propagation of every combination of the one-hot features through the neural network. This makes reachability analysis infeasible for more than a moderate number of one-hot features, as if there are 11 one-hot features, each with 10 different options, there are 10^{11} different combinations that must be handled.

3) Vertex Enumeration: Our method performs vertex enumeration by computing the convex-hull of the dual of the constraints. This is fast enough for the limited number of dimensions on which we try our method, as currently the limiting factor is the exponential growth of the size of the discretized grid. However, it still bears a significant time penalty and this method can be improved by using the Avis-Fakuta method, which runs in linear time on the number of vertices.

4) Polytope Volume Computation: QHULL is used to compute the volume of the Polytope due to its accessibil-

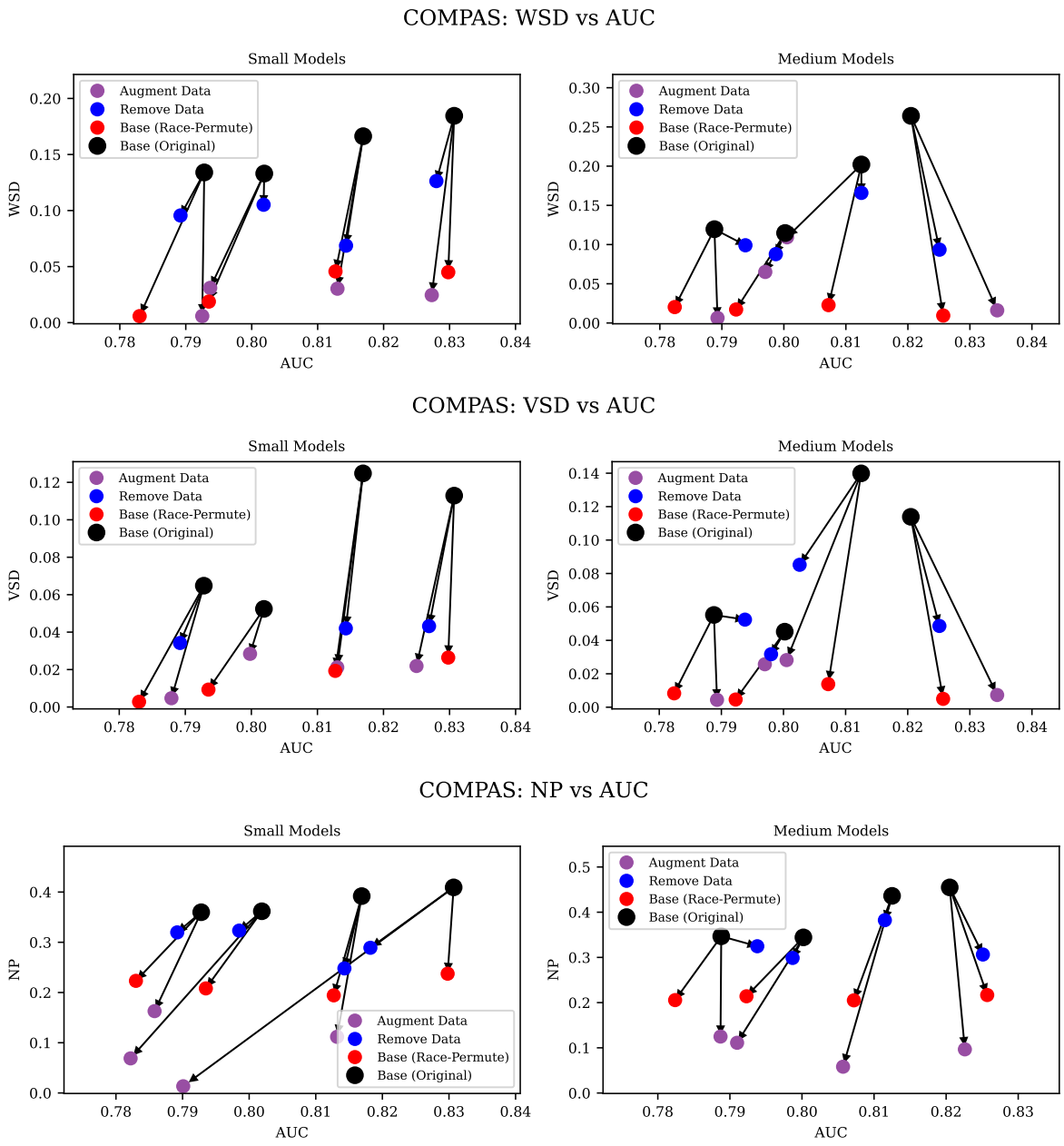

Fig. 2. COMPAS Dataset: Model performance/AUC vs Model fairness/NP under three fairness-sensitive training methods. Arrows down and to the left reflect models that are fairer but less accurate than the original mode. We obtain large improvements in fairness at little cost in accuracy.

ity and robustness. However, as the input dimensionality and the number of vertices grows, this becomes suboptimal. Lawrence's algorithm runs in linear time in the number of vertices, providing a large potential speed-up. An experiment demonstrating the potential improvement in scalability based on the polytopes computed during our analysis is shown in figure (4).

5) Explicit Encoding of Classes: Our analysis methods assumes that the classes analyzed can be represented as Polytopes in the input-space. This limits the applicability of the method to analyzing fairness of latent variables. However, if one has an estimate for the form of the latent variable, this can be approximated with a union of polytope in the input space and analyzed.

V. Conclusion

We introduce a method to formally analyze the fairness of neural network models. Our fairness analysis methods does not require access to the model's training data, enabling model fairness evaluation in a broad range

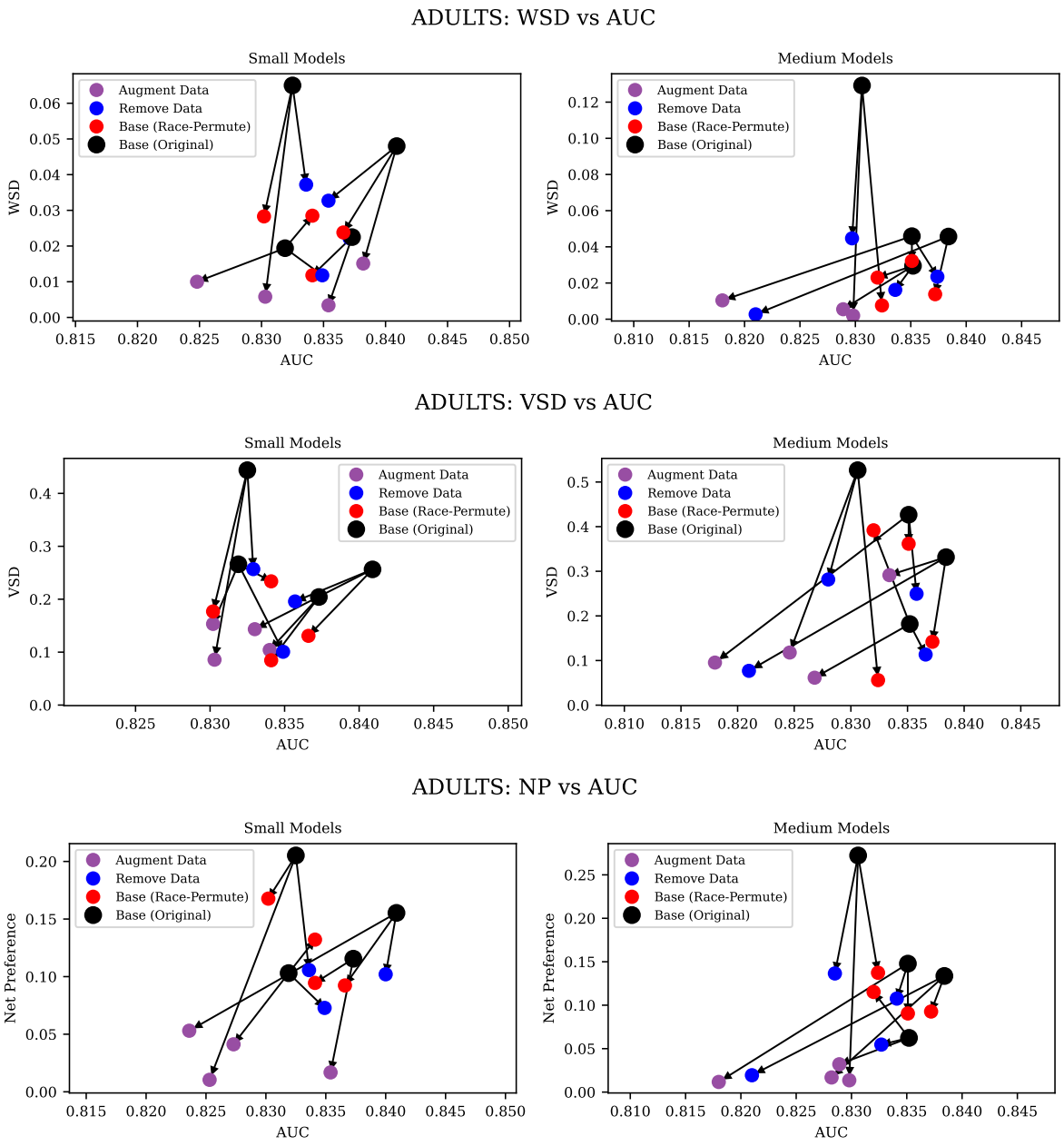

Fig. 3. ADULTS Dataset: Model performance/AUC vs Model fairness/NP under three fairness-sensitive training methods. Arrows down and to the left reflect models that are fairer but less accurate than the original mode. We obtain large improvements in fairness at little cost in accuracy.

of contexts. We use our method to evaluate the fairness of several models, and demonstrate that without intervention trained models often demonstrate a significant degree of unfairness. We demonstrate that for circumstances in which the probability distribution on the inputs is not accessible, a probability-free approach provides a suitable substitute. Our results show that our data-augmentation method is a simple yet effective method for improving the model's fairness. Critically, it is also effective at reducing reliance on proxy variables of the protected class.

We explored the limits of scalability of our method, showing that there is still much room to improve runtime performance by replacing the algorithms used the enumerate polytope vertices and to find the volume of the polytope from these vertices with more optimal ones.

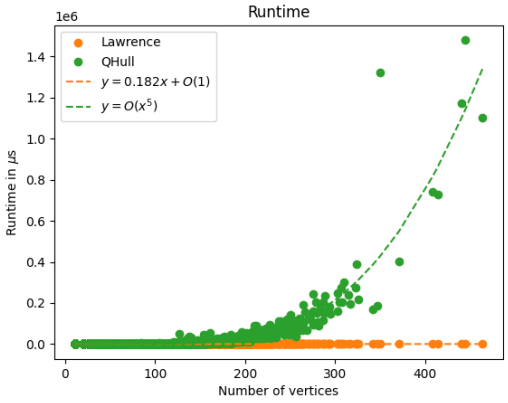

Fig. 4. Comparison of QHull and Lawrence's Algorithm for Volume Computation (10 Constraints in 10D Space)

References

[1] A. Albarghouthi. *Introduction to Neural Network Verification*. verifieddeeplearning.com, 2021. http://verifieddeeplearning.com.

[2] S. Bak, C. Liu, and T. T. Johnson. The second international verification of neural networks competition (VNN-COMP 2021): Summary and results. *CoRR*, abs/2109.00498, 2021.

[3] S. Bak, H.-D. Tran, K. Hobbs, and T. T. Johnson. Improved geometric path enumeration for verifying ReLU neural networks. In *32nd International Conference on Computer Aided Verification*. Springer, 2020.

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

[5] R. K. Bellamy, K. Dey, M. Hind, S. C. Hoffman, S. Houde, K. Kannan, P. Lohia, J. Martino, S. Mehta, A. Mojsilovic, et al. Ai fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias. *arXiv preprint arXiv:1810.01943*, 2018.

[6] S. Caton and C. Haas. Fairness in machine learning: A survey. *arXiv preprint arXiv:2010.04053*, 2020.

[7] P. S. Duggirala and M. Viswanathan. Parsimonious, simulation based verification of linear systems. In *International Conference on Computer Aided Verification*, pages 477–494. Springer, 2016.

[8] I. Z. Emiris and V. Fisikopoulos. Practical polytope volume approximation. *ACM Trans. Math. Softw.*, 44(4), jun 2018.

[9] S. Galhotra, Y. Brun, and A. Meliou. Fairness testing: testing software for discrimination. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, pages 498–510, 2017.

[10] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. Vechev. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2018.

[11] R. Ivanov, J. Weimer, R. Alur, G. J. Pappas, and I. Lee. 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*, pages 169–178, 2019.

[12] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*. Springer, 2017.

[13] C. Liu, T. Arnon, C. Lazarus, C. Strong, C. Barrett, and M. J. Kochenderfer. Algorithms for verifying deep neural networks. *arXiv preprint arXiv:1903.06758*, 2019.

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

[15] Y. Meng, Z. Qiu, M. T. B. Waez, and C. Fan. Case studies for computing density of reachable states for safe autonomous motion planning. In *NASA Formal Methods Symposium*, pages 251–271. Springer, 2022.

[16] Y. Meng, D. Sun, Z. Qiu, M. T. B. Waez, and C. Fan. Learning density distribution of reachable states for autonomous systems. In *Conference on Robot Learning*, pages 124–136. PMLR, 2022.

[17] A. Ruoss, M. Balunović, M. Fischer, and M. Vechev. Learning certified individually fair representations. *arXiv preprint arXiv:2002.10312*, 2020.

[18] H.-D. Tran, S. Bak, W. Xiang, and T. T. Johnson. Verification of deep convolutional neural networks using imagestars. In *Proceedings of the 32nd International Conference on Computer Aided Verification*. Springer, 2020.

[19] H.-D. Tran, F. Cai, M. L. Diego, P. Musau, T. T. Johnson, and X. Koutsoukos. Safety verification of cyber-physical systems with reinforcement learning control. *ACM Transactions on Embedded Computing Systems (TECS)*, 18(5s):1–22, 2019.

[20] H.-D. Tran, W. Xiang, and T. T. Johnson. Verification approaches for learning-enabled autonomous cyber-physical systems. *IEEE Design & Test*, 2020.

[21] C. Urban, M. Christakis, V. Wüstholz, and F. Zhang. Perfectly parallel fairness certification of neural networks. *Proceedings of the ACM on Programming Languages*, 4(OOPSLA):1–30, 2020.

[22] J. A. Vincent and M. Schwager. Reachable polyhedral marching (rpm): A safety verification algorithm for robotic systems with deep neural network components. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9029–9035. IEEE, 2021.

[23] C. Wadsworth, F. Vera, and C. Piech. Achieving fairness through adversarial learning: an application to recidivism prediction. *arXiv preprint arXiv:1807.00199*, 2018.

[24] W. Xiang, P. Musau, A. A. Wild, D. M. Lopez, N. Hamilton, X. Yang, J. Rosenfeld, and T. T. Johnson. Verification for machine learning, autonomy, and neural networks survey. *arXiv preprint arXiv:1810.01989*, 2018.

[25] W. Xiang, H.-D. Tran, X. Yang, and T. T. Johnson. Reachable set estimation for neural network control systems: A simulation-guided approach. *IEEE Transactions on Neural Networks and Learning Systems*, 32(5):1821–1830, 2020.

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

TABLE II

MODEL PERFORMANCE(*AUC*) *v.s* MODEL FAIRNESS OVER 4 RUNS FOR THE SMALL/MEDIUM MODELS ON THE ADULTS AND COMPAS DATASETS. FAIRNESS-SENSITIVE TRAINING METHODS (RACE PERMUTATION, AND DATA REMOVAL/AUGMENTATION METHODS) REDUCE UNFAIRNESS BY AN AVERAGE OF 65.1% WHILE REDUCING MODEL QUALITY (AUC) BY LESS THAN 1%.

			COMPAS				ADULTS			
Model Size	Trial ID	Measure	Baseline	Perm Race	Rem Data	Aug Data	Baseline	Perm Race	Rem Data	Aug Data
Small	0	AUC	**0.8169**	0.8127	0.8143	0.8130	**0.8373**	0.8341	0.8371	0.8354
		WSD	0.1663	0.0457	0.0687	**0.0303**	0.0225	0.0118	0.0222	**0.0034**
		VSD	0.1248	**0.0193**	0.0420	0.0212	0.2042	**0.0846**	0.2042	0.1041
		NP	0.3919	0.1943	0.2480	**0.1118**	0.1155	0.0945	0.1155	**0.0168**
	1	AUC	**0.7928**	0.7830	0.7892	0.7925	0.8319	0.8341	**0.8349**	0.8248
		WSD	0.1340	**0.0058**	0.0956	0.0059	0.0194	0.0285	0.0118	**0.0100**
		VSD	0.0648	**0.0028**	0.0342	0.0047	0.2661	0.2338	**0.1007**	0.1533
		NP	0.3598	0.2232	0.3197	**0.1629**	0.1030	0.1321	0.0728	**0.0412**
	2	AUC	**0.8019**	0.7935	0.8018	0.7937	**0.8409**	0.8366	0.8354	0.8382
		WSD	0.1330	**0.0188**	0.1052	0.0310	0.0480	0.0238	0.0327	**0.0151**
		VSD	0.0524	**0.0093**	0.0517	0.0284	0.2564	**0.1308**	0.1958	0.1434
		NP	0.3617	0.2082	0.3232	**0.0689**	0.1553	0.0923	0.1020	**0.0530**
	3	AUC	**0.8307**	0.8298	0.8280	0.8273	0.8325	0.8302	**0.8336**	0.8303
		WSD	0.1845	0.0450	0.1263	**0.0246**	0.0650	0.0283	0.0372	**0.0058**
		VSD	0.1129	0.0264	0.0432	**0.0219**	0.4439	0.1769	0.2569	**0.0857**
		NP	0.4093	0.2374	0.2891	**0.0134**	0.2052	0.1677	0.1057	**0.0104**
	Avg	AUC	**0.8106**	0.8047	0.8083	0.8066	**0.8357**	0.8337	0.8353	0.8322
		WSD	0.1545	0.0288	0.0990	**0.0230**	0.0387	0.0231	0.0260	**0.0086**
		VSD	0.0887	**0.0144**	0.0428	0.0191	0.2927	0.1565	0.1894	**0.1216**
		NP	0.3807	0.2158	0.0990	**0.0230**	0.1448	0.1217	0.0260	**0.0086**
Med.	0	AUC	**0.8125**	0.8072	0.8125	0.8005	**0.8384**	0.8372	0.8210	0.8384
		WSD	0.2021	**0.0228**	0.1659	0.1093	0.0457	0.0138	**0.0027**	0.0457
		VSD	0.1399	**0.0138**	0.0852	0.0282	0.3317	0.1420	**0.0769**	0.2913
		NP	0.4361	0.2050	0.3823	**0.0582**	0.1336	0.0927	0.0194	**0.0168**
	1	AUC	0.7888	0.7824	**0.7938**	0.7893	**0.8352**	0.8320	0.8336	0.8289
		WSD	0.1196	0.0203	0.0991	**0.0064**	0.0295	0.0230	0.0162	**0.0055**
		VSD	0.0552	0.0083	0.0524	**0.0044**	0.1818	0.3919	0.1134	**0.0615**
		NP	0.3467	0.2055	0.3247	**0.1249**	0.0624	0.1153	0.0547	**0.0320**
	2	AUC	**0.8002**	0.7923	0.7987	0.7970	0.8351	0.8351	**0.8374**	0.8180
		WSD	0.1147	**0.0172**	0.0878	0.0651	0.0459	0.0322	0.0235	**0.0104**
		VSD	0.0452	**0.0046**	0.0317	0.0257	0.4267	0.3616	0.2495	**0.0954**
		NP	0.3443	0.2140	0.2988	**0.1111**	0.1478	0.0905	0.1076	**0.0117**
	3	AUC	0.8205	0.8257	0.8251	**0.8344**	0.8306	**0.8324**	0.8297	0.8298
		WSD	0.2641	**0.0095**	0.0935	0.0161	0.1292	0.0076	0.0447	**0.0020**
		VSD	0.1139	**0.0050**	0.0486	0.0073	0.5266	**0.0560**	0.2816	0.1179
		NP	0.4549	0.2168	0.3064	**0.0966**	0.2723	0.1373	0.1365	**0.0136**
	Avg	AUC	0.8055	0.8019	**0.8075**	0.8053	**0.8348**	0.8342	0.8304	0.8288
		WSD	0.1751	**0.0175**	0.1116	0.0492	0.0626	0.0192	0.0218	**0.0159**
		VSD	0.0885	**0.0079**	0.0545	0.0164	0.3667	0.2379	0.1804	**0.1415**
		NP	0.3955	0.2103	0.1116	**0.0492**	0.1540	0.1089	0.0218	**0.0159**