Global Interpretation Weighted Median Value Replacement for Mitigating Sensitive Attribute Leakage in Neural Networks

Amit Joshi
Department of Computer Science
University of Texas, Austin
Austin, USA
amitjoshi24@utexas.edu

Nikhil Ajjarapu
Department of Computer Science
University of Texas, Austin
Austin, USA
nikhil.ajjarapu@utexas.edu

Priyal Belgamwar
Department of Computer Science
University of Texas, Austin
Austin, USA
priyal.belgamwar@utexas.edu

***Abstract*—From housing and car loan approval, to criminal sentencing, to voice and facial recognition, neural networks have proliferated greatly over the last few years to power the next generation of technological advances. As they grow in popularity, however, it becomes just as important to ensure that everyone, including marginalized groups, has equal access to these technologies. Without careful oversight on the behalf of developers, neural networks have the potential to amplify existing biases. One of the more common methods of bias found in neural networks that have protected attributes in their training data is when neurons corresponding to non-sensitive input features become highly correlated to sensitive input features after feature propagation, a phenomenon known as *sensitive attribute leakage*. In this paper, we attempt to mitigate the effects of sensitive attribute leakage by defining and utilizing a post-processing technique named weighted median value replacement to modify the neuron weights in a post-processing step. Using equalized odds as a measure of fairness, we measure the tradeoff between fairness and accuracy when implementing weighted median value replacement. We find that weighted median value replacement is able to significantly increase fairness in an already trained neural network without compromising accuracy, demonstrating its potential as a model-agnostic method to improve fairness in neural networks with minimal external effort.**

***Index Terms*—neural networks, fairness, bias sensitive attribute leakage, weighted median value replacement**

1. Introduction

Often, as society has progressed technologically, the benefits of such developments are not felt equally - those in positions of privilege tend to be the ones to enjoy most of the benefits. As machine learning and neural networks take a step forward in capability and become more integrated into daily society, it becomes increasingly important that we ensure these algorithms do not leave marginalized subgroups behind in their wake. When we rely on heavily data based models that require lots of real life examples to make meaningful predictions, there is a strong risk of reifying and entrenching already-existing biases in society today. In order to avoid this, it is key that algorithms and datasets are consciously built in a manner consistent with equal representations for all relevant subgroups even if it comes with the tradeoff of lower accuracy metrics. However, it is important to define what "fairness" clearly means in this context if we are to integrate this concept mathematically into our algorithms. Hardt et al. in 2016 suggested fairness to mean the general principle of "equal opportunity by design" [1] (the phrase was borrowed from an Obama administration report regarding discrimination in automated learning). This means there needs to be a calculated decision made regarding the data used to train algorithms, and careful monitoring of outcomes to ensure that the algorithm is not making biased decisions. It is important to have a general understanding of the different forms of unfairness and bias that can arise in these situations as well.

1. Unfairness in Neural Networks

Bias and unfairness in datasets can take many different forms, but there are two broad categories that most situations can be sorted into: prediction outcome discrimination and prediction quality disparity. Prediction outcome discrimination refers to discrimination that occurs as a result of being a member of a certain protected subgroup, whereas prediction quality disparity refers to discrimination that occurs as a result of lower model quality (due to underrepresentation in the dataset, for example) for certain protected subgroups. In this paper, we will focus on prediction outcome discrimination, of which there are two further subtypes of bias: discrimination via representation and discrimination via input, one of the most common sources of bias in neural networks and our main focus in this paper. Discrimination via input happens when the training data is biased in some way or other against a certain subgroup, and usually causes "resources and opportunities allocations harm in high-stake applications" [2], such as employment and loan approval contexts. Popular examples of this include Amazon's internal employment algorithm that learned to bias against applications from women, or Northpointe's criminal recidivism algorithm that was shown to be biased against black men when predicting rates of reoffending for people convicted of a crime for judges to use in making a decision regarding prison sentencing. In all of these cases, the algorithm utilized a protected attribute (race, gender, etc.) as part of the decision making process in a biased manner (e.g not enough women employees at Amazon leading the algorithm to believe men are more desirable as candidates). Thus, discrimination via input can be succinctly defined as "amplifying societal stereotypes by over associating protected attributes, e.g., race and gender, with the prediction task" [2]. This over-association can happen even if the input data to the algorithm does not contain the sensitive attribute directly. While input data usually does have sensitive attribute information, reasons to remove a protected attribute entirely include privacy protection laws (e.g medical data), the nature of the input data especially if the algorithm is dealing with raw unprocessed text or image data where the sensitive attribute is not obvious, or if we are trying to achieve fairness through the concept of "fairness through unawareness" [1], which is defined as "ignore all protected attributes such as race, color, religion, gender, disability or family status" [1]. In all of these cases, the algorithm can regardless quickly form biased intermediate representations due to an effect called **sensitive attribute leakage**.

2. *Sensitive attribute leakage*

Sensitive attribute leakage refers to when sensitive attributes in the input data become highly correlated with non-sensitive attributes, which allows the algorithm to use these highly correlated features as a proxy for the original sensitive attribute when making predictions on new data. A more technical definition of sensitive attribute leakage is as follows: "[S]ome original innocuous feature channels that have lower correlation to sensitive channels and encode less sensitive information may become highly correlated to sensitive ones after feature propagation and hence encode more sensitive information, which is termed as sensitive attribute leakage" [3]. This biases intermediate representations formed in the algorithm due to biases in the training data, and given that neural networks have low interpretability, this compounds the problem as it is often hard to know why the network is making the prediction the way it is. For example, if we are dealing with housing data in low-income neighborhoods with race as a protected attribute, the feature "zip code" could become highly correlated with the sensitive attribute "race" (due to societal effects such as redlining that cluster minorities together-), and zip code would serve as a proxy for race which would cause the algorithm to become over-reliant on race as a predictive feature, which in turn leads to biased predictions. Thus, combating sensitive attribute leakage is an important step in ensuring a fair and unbiased neural network, even if it comes at the cost of overall algorithmic accuracy.

3. *Our proposal*

Our goal for this project was to develop a method that could mitigate sensitive attribute leakage, either as a pre-processing step (modify training process before the models are trained) or a post-processing step (modify the model after it is trained and finalized). In this paper, we propose a method to debias inputs (given a sensitive attribute s) of an already-trained neural network using **weighted median value replacement** to modify neuron inputs (for the input layer) to be closer to the median of corresponding (same percentile) neuron values over different classes of s during evaluation. In essence, by reducing the numerical difference between the neuron inputs most correlated with sensitive attributes, we hope to reduce the amount of leakage occurring between sensitive and non-sensitive features within the model. This can be done using *global interpretability* methods that allow us to see which neurons were relevant in predicting the sensitive attribute, ultimately allowing us to see how the neural network prioritizes non-sensitive features that nevertheless correlate with sensitive features. These changes will be implemented as a post-processing step, as we are modifying the neuron values after the model is trained, which ensures the methods are far more model-agnostic than as a pre-processing step. However, we also additionally explore the effect of retraining a neural network on debiased training input data and evaluating on debiased test input data, which is indeed preprocessing. Ultimately, we will measure the effect these changes have on the overall "fairness" of the trained neural network, while monitoring the tradeoff being made between the fairness and accuracy of the network. This can be done by partitioning the data into the two labels (is_recid) and looking at the model outputs for each sensitive group. We decided to evaluate the fairness of our models with the equalized odds metric, which we expand upon below.

4. *Defining fairness - demographic parity vs equalized odds*

As discussed previously, the definition of what fairness is exactly has several schools of thoughts, ranging from "fairness through unawareness" to "equal opportunity by design". There are a variety of mathematical analogues to these concepts when deciding how to define our objective function for fairness. **Demographic parity** corresponds to the concept of "fairness through unawareness", and it "asserts that [the] average of algorithmic decisions should be similar across different groups…independent of the ground truth labels" [2]. In other words, given a binary classification task, the positive rate for the model should be equal across all subgroups of the sensitive attribute, as seen in the illustration in Figure 1.

However, the issue with this definition is there is no measure of "quality" for the final outcomes per subgroup. To ground this argument with a real world example, if our model is classifying 8 loan applicants for mortgage approval with the sensitive attribute being race, it could theoretically approve 4 high risk black applicants (due to class imbalance in the dataset or poorer quality data for black applicants) and 4 low risk white applicants, and both subgroups would have a positive rate of 50% (4/8), seemingly satisfying the conditions for demographic parity. This intuitively should not hold however, as the model is performing much more poorly on black applicants compared to white applicants. **Equalized odds** instead focuses on true and false positives: the model must have the same true and false positive rates for each subgroup of the sensitive attribute. A true positive rate can be thought of as the probability of the model outputting 1 given that the

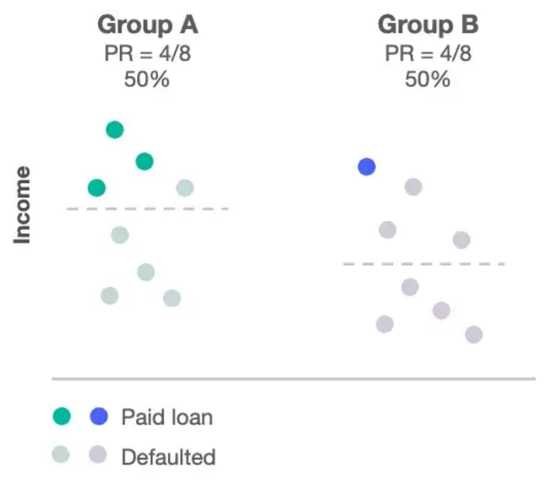

Fig. 1. Demographic Parity

true label is 1, and the false positive rate can be thought of as the probability of the model outputting 1 given that the true label is 0. The above loan approval example would not work under this new measure, as the model would have a true positive rate of 0% for black applicants and 100% for white applicants. Equalized odds can be seen depicted graphically in Figure 2.

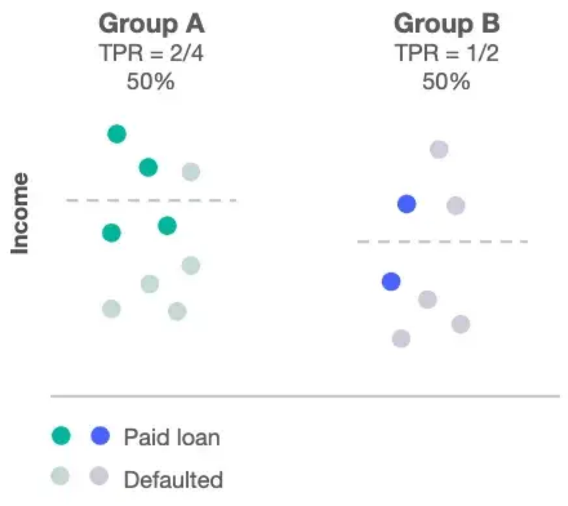

Fig. 2. Equalized Odds

Thus, in this project we used the equalized odds measure to see how well our model performed on our test data, as it provides a more holistic view of how the model performs across protected subgroups.

5. Global Interpretation

While we will go into this in more detail below, we use global interpretation methods to help understand how the model is making decisions and what features it prioritizes when doing so. We specifically find the feature attribution scores of the model with respect to given samples, where attribution scores correspond to how important a input feature was when generating the final output. We do this through a method called Integrated Gradients, which is a versatile technique that works on any differentiable model. Integrated Gradients works by computing the gradient of the model function with respect to each of the input features, and measuring how the gradient changes when computed across a linear interpolation of input data from a given baseline to any data point in the input dataset. This technique allowed us to get a basic understanding of how the model is making decisions, which allowed us to determine whether our model was becoming more fair or not.

2. Background and Related Work

Related work in the area has dealt with mitigating sensitive attribute leakage in three broad categories - preprocessing, inprocessing, and postprocessing. After a thorough literature review, we took ideas from all three areas for this paper. *Feldman 14* [6] deals specifically with preprocessing. They aim to remove disparate impact (a measure of fairness) from the dataset by an algorithm similar to our weighted median value replacement algorithm. In particular, the geometric repair algorithm proposed in their paper works as follows:

- For each sensitive group, and for each feature, create a percentile/value mapping e.g. If the feature was “height”, and the sensitive group was “male” (with the sensitive attribute being “sex”), the percentile/value mapping can tell that the 25^{th} percentile for height for males is 5’7”.
- Given a sample, and a feature value:
  - Calculate percentile of value among its own sensitive group
  - Calculate median of corresponding (same percentile) values for the same percentile over all sensitive groups
  - Replace original value with a weighted average (λ) of median and original value
- Repeat for all samples and features

Their approach works quite well, achieving a great balance between fairness and accuracy. Looking at their algorithm, we wondered if it was necessary to median out all input features to the same degree. In this paper, we modify their geometric repair algorithm by dynamically calculating the degree to which each feature’s values should be modified (with each feature i having it’s own λ_i), using weights provided by Global Interpretation.

We also took ideas from papers we read about in-processing. *Du 2020* talks about the importance of only using certain features in predicting certain outcomes [2]. These are given via “feature-wise annotations” to determine which features are fair and/or relevant in making certain decisions. These annotations can be given by domain experts or by using counterfactuals (generated by seeing how perturbations of input affect model output). Using other features added an extra penalty in the loss function [7] [8]. More similar work to ours uses Local interpretation [9] (using attention to highlight the relevant parts of the input) to attribute predictions to specific features. We use Global Interpretation on our sensitive model (DNN that predicts the sensitive attribute) to see which features were used in this process. Ideally, the sensitive model shouldn't be able to predict the sensitive attribute because this indicates that our target model (DNN that predicts the main objective) is encoding the sensitive attribute in its hidden representations.

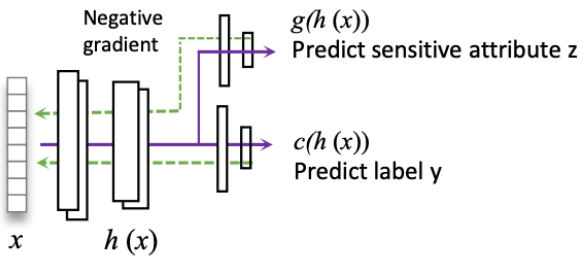

Fig. 3. Adversarial Learning

We also took ideas from adversarial learning, where the target model (predictor) and the sensitive model (adversarial classifier) are trained simultaneously. The predictor aims to learn a representation that is maximally informative for its predictive task while the adversarial classifier aims to minimize the ability to predict the sensitive attribute [2]. The predictor is denoted as $f(x) = c(h(x))$, where $h(x)$ is the intermediate representation (generated by the predictor) for input x, and $c(y)$ is responsible to map intermediate representation to the final model prediction. The protected attribute is denoted using z. An adversarial classifier $g(h(x))$ is also constructed to predict sensitive attribute z from representation $h(x)$. A drawback of this approach is that this may harm model accuracy and training can be unstable. We propose to keep the structure of having a sensitive model which predicts the sensitive attribute from the hidden representation of the target, but not to train the two simultaneously to combat the training instability issue. Instead, we train g (final layer of sensitive model) after training the target model. This way, h (part of both target and sensitive models that generates hidden representation) and c (part of target model that takes in its hidden representation via h and predicts the target) have finished training and hence the weights corresponding to h are frozen while training the sensitive model, as h is shared between the target and sensitive model. Now, we can use Global Interpretation on the sensitive model to see which inputs to the shared component h were relevant in predicting the sensitive attribute.

Adversarial learning tries to ensure that information relating to the sensitive attribute is not available from the hidden representation by using an adversarial classifier to attempt to do just that - predict the sensitive attribute and use its performance as a measure of how much information the hidden representation has about the sensitive attribute. Another approach seen is to use distance metrics between hidden representations of samples with different sensitive attributes and ensuring that they are small, by penalizing these distances in logistic regression [10] for instance. This gives weight (no pun intended) to the notion that the hidden representations of different sensitive groups should not encode sensitive information and fairness will improve if these representations do not encode sensitive information.

3. Implementation and Architecture Details

1. *Target Model*

We first begin by loading the dataset and eliminating the sensitive attribute from the training data before the preprocessing steps. We thus obtain a dataset to predict the label void of the targeted input feature which is considered sensitive.

We proceed to train a simple deep neural network to predict the main target, which we will refer to as the **Target Model**. Here, we try to obtain considerable accuracy based on the dataset chosen. The number of layers and the number of neurons in each layer are decided based on the number of input features and the size of the dataset, along with other modulations of the dataset. Following that, we evaluate the neural network on certain metrics to gauge the model performance and fairness.

2. *Fairness metrics: True Positive and False Positive Parity*

Our goal is to satisfy equalized odds, but that is difficult to exactly attain in practice. So, we decided to measure how far away a model is from achieving equalized odds. First, we considered true positive parity, which is a measure of how different true positive rates are across sensitive groups, and likewise for false positive parity and false positive rates. For our binary classification purposes, and with two sensitive groups, we have the following:

R: Model Output
Y: True Label
A: Sensitive Attribute
a, b: Sensitive Values

True Positive Parity:
$|P(R = 1|Y = 1, A = a) - P(R = 1|Y = 1, A = b)|$
False Positive Parity:
$|P(R = 1|Y = 0, A = a) - P(R = 0|Y = 1, A = b)|$

3. *Sensitive Model*

We generate a second dataset in which we modify the original dataset by making the sensitive attribute (which was initially omitted) the dependent variable. Here we intend to predict the sensitive feature using the sample input features from the original dataset, thus keeping the same dimensions of the dataset and pulling out the original output label.

Next, we create a second neural network to operate on this dataset, which we will refer to as the **Sensitive Model**. The purpose of this neural network is to now predict the sensitive feature based on the input features in absence of the primary label, thus finding the attributes which are most correlated to the sensitive attribute. The structure of this neural network is identical to the Target model (the original neural network). Additionally, it takes in hidden representations outputted by the penultimate layer from the primary neural network. The weights and biases shared from the Target model are frozen during training to suspend weight/bias updates on the penultimate layers. The training takes place on the second dataset, where only the last layer of the secondary neural network is trained to predict the output, which is the sensitive attribute in this case. Backpropagation training occurs solely on the last layer.

At first, we found that the sensitive model just trivially outputted the majority sensitive value for every prediction. To combat this, we used class weighted binary cross entropy loss, weighted by the inverse frequency of each class in the training dataset:

$$\mathcal{L}_{\text{BCEwt}} = \frac{1}{N}\sum_{i=1}^{N} \beta_1 y_i log(ypred_i) + \beta_0(1-y_i)log(1-ypred_i)$$

$$\beta_1 = 1 - \left(\frac{\sum_{i=1}^{N} \mathbb{1}_{y_i=1}}{N}\right)$$

$$\beta_0 = 1 - \left(\frac{\sum_{i=1}^{N} \mathbb{1}_{y_i=0}}{N}\right)$$

4. Feature Importance Attributions

We propose to adopt methods of global interpretation in order to holistically explain the behavior of our neural network. In this approach, we start with identifying the input features that are most correlated to the selected sensitive attribute. For this purpose, we find the attribution scores with respect to the inputs of the model. To maintain a global view of the feature importances, the attributions are aggregated across the entire test dataset. These attribution scores are used in the consequent stage for modifying the input values. To calculate these scores, we use the **Integrated Gradients** technique. Sundararajan 17 [5] developed the integrated gradients approach to satisfy two axioms related to attributions. The first axiom states that for two inputs that result in different predictions and that differ in just a single feature, the attribution for that feature should be non-zero. The second axiom states that two networks should have the same attributions if they have the same outputs for all values of inputs. The integrated gradients approach uses the path integral of the gradient to find out the activations of all feature dimensions.

5. Creation of Medianed Test Dataset

We further proceed by using these attribution values to apply sensitive weight medianing to the input features. The goal is to create a debiased test dataset by modifying each input neuron value to be closer to the median of corresponding input neuron values, or neurons having the same percentile over the sensitive attribute. However, some inputs should be closer to their median values - namely the ones with high feature importance to predict the sensitive attribute. This can be done using **Weighted Median Value Replacement**.

1) Weighted Median Value Replacement: The weighted median value replacement algorithm is similar to the geometric repair algorithm in [6], except that each feature has its own replacement weight - a parameter in the range $[0, 1]$ which represents a weighted average of the median corresponding feature value (of the same percentiles over all sensitive groups) and the current sample's feature value.

We found that higher replacement weights (closer to the median) perform better, so at minimum we take 0.5 of the median value, known as λ_{base}, which is case dependent (models/datasets with low amounts of bias may benefit from a lower λ_{base}). The replacement weight for each feature (λ_i) is scaled linearly with the magnitude of its feature importance (α_i) (given by Global Interpretation) and the maximum magnitude feature importance. So, our final replacement weight is given by:

$$\lambda_i = \lambda_{base} + (1 - \lambda_{base}) * \left(\frac{|\alpha_i|}{max_j|\alpha_j|}\right) \quad (1)$$

2) Optimization for features originating from a small space: It is also unclear in [6] how the percentile/value mapping is calculated, which could especially be an issue for features with relatively few values. For example, what is the percentile of 2 in $[1, 2, 2, 2, 2]$? It is both the largest element and the second-lowest element. To combat this, we propose a logical way of calculating what the percentile should be. We accomplished this using two slightly different binary searches. The first is a lower binary search, which finds the lowest index of the specified value occurring (or the largest element less than the specified value in the case that the specified value doesn't appear in the list). In the previous example, this would be index 1 (0-indexed array) for element 2. The second is an upper binary search, which naturally finds the highest index of the specified value occurring (or the smallest element greater than the specified value in the case that the specified value does not appear in the list). In the previous example, this would be index 4 for element 2. Then, the average of these two values is taken to obtain the index and percentile of the given feature value among its own sensitive group.

The Target Model is tested on the debiased test dataset and we evaluate the prediction results based on the fairness metrics specified above. We finally compare the initial fairness of the model with the fairness attained on the same model using the debiased test data.

6. Retraining

After getting satisfactory results on the medianed test dataset, we used the same scores and methods to create another dataset for retraining. This constitutes a new medianed train dataset where all the input features are modified based on the attribution scores and dynamic medianing of the training dataset. Hereby, using postprocessing steps on the original primary model, we are also able to develop a novel preprocessing technique to be applied to the dataset before training a new model. The primary neural network is retrained on this new training data and then evaluated on the medianed test data. The same metrics for accuracy and fairness are used to evaluate the new model and we compare it with the primary model results.

4. Experimental Design and Results

1. *COMPAS Dataset*

We tried our approach on the COMPAS (Correctional Offender Management Profiling for Alternative Sanctions) dataset, developed by Northpointe. COMPAS is a popular commercially used algorithm to score criminal defendants based on their likelihood to re-offend or recidivate. It was shown by ProPublica that the COMPAS algorithm is biased against black defendants as compared to white defendants, where the former were predicted to have a higher risk of recidivism than the true risk and hence were more than twice as likely to be misclassified as high risk as compared to their white counterparts.

We chose to use this dataset for our experimental purposes to investigate the capability of our proposed method to detect racial bias from our primary model (Target Model) and the ability to mitigate this bias found in the original model.

2. *Target Model*

We first load the COMPAS dataset which contains the various features of criminal defendants including age, sex, juvenile felony count, priors count, and race. We filter out the observations such that the "Race" feature only has the values of Caucasian (white) and African-American (black) so as to base the bias solely on these two races for analysis. We then eliminated the "Race" feature from the dataset and use the rest of the features to predict the recidivism score.

We create a deep neural network to make recidivism predictions, which would be our Target Model. The structure of the neural network contains an input layer, three fully connected hidden layers and an output layer, with a LeakyReLU activation function between each layer. Each hidden layer contains 50 neurons, which we found as the optimal number for attaining satisfactory accuracy.

To visualize fairness, let us take a look at separation graphs. Separation graphs divide the data into their true labels, and then see the distributions of different sensitive values within the same true label. The separation graph for the original Target Model can be seen in Figure 4. It is evident from the graph that the Target Model has some information about race, as black people have significantly higher scores than white people, even when the true target label *is_recid* is the same.

The Target Model was then evaluated based on the fairness metrics mentioned previously. We trained for 15 epochs and achieved an accuracy of 67.614%. The accuracy and fairness metrics can be seen in Tables 1 and 2.

3. *Sensitive Model*

We generate a second dataset, as previously described in Section 3.3, by changing the output label to *Race* instead of *Is_Recid*, keeping the input features and dimensions intact.

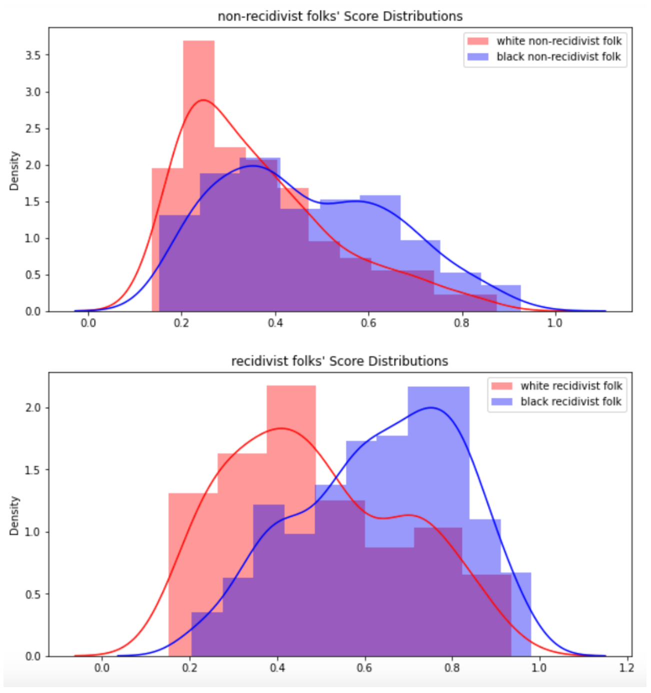

Fig. 4. Original Target Model Separation Graph

Thus, the data is set to predict the race using the same attributes.

The Sensitive Model (secondary neural network) is created to predict the race instead of the recidivism score. The structure of the neural network is the same as that of the Target Model, with the input and hidden layers having the same structure and weights. The borrowed weights are frozen for training and only the weights of the last layer are updated using backpropagation as depicted in Figure 5. The Sensitive Model was able to achieve an accuracy of about 66% suggesting that the original model was heavily dependent on race through redundant encodings for making recidivism predictions.

The basis for the Sensitive Model can be seen in Figure 4, which show our Target Model's outputs among the 4 groups of 2 target values $\times$ 2 sensitive values.

As discussed before, the original Target Model is significantly biased. A rudimentary sensitive model can even use the same last layer as the target model and simply assign *black* to those assigned *is_recid* and get nontrivial accuracy, exemplifying a large amount of bias in the model. Although in our paper the last layer of the sensitive model is indeed different from the last layer of the target model.

4. *Feature Importance Attributions*

To explain the behavior of our neural network, we identify the input features that are most important in predicting the race attribute, thus giving us an insight into the features that are the redundant encodings for the sensitive attribute race. To this end, we find the attribution scores with respect to the Sensitive Model using the Integrated Gradients technique (elaborated in Section 3.4). These scores can be seen in the Figure 6.

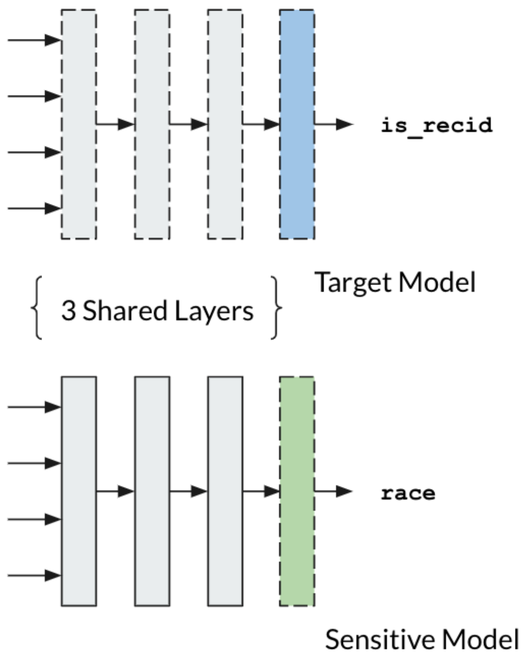

Fig. 5. Neural Network Architectures

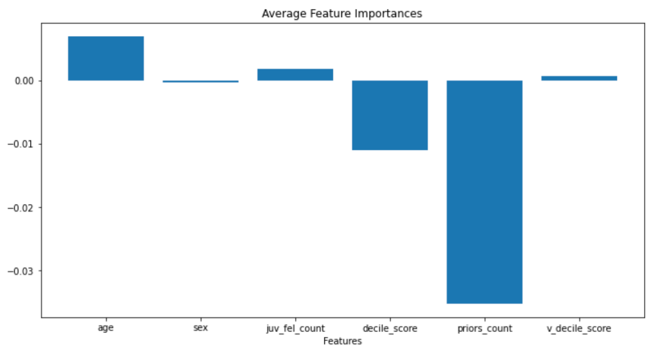

Fig. 6. Feature Attribution Scores

5. *Medianed Test Dataset*

The attribution values obtained using Integrated Gradients are used to replace input weights using weighted median (described in Section 3.5) to create a debiased test dataset. The degree of modification depends on the relative importance of the respective features apropos the sensitive attribute– race. We then test the Target Model on this new debiased test dataset and evaluate the fairness of the predictions with respect to race.

We achieved the accuracy and fairness as illustrated in Table 1 and 2. We finally compare the initial fairness of the Target Model with the fairness attained on the same model using the debiased test data and deduce inference about racial bias before and after the application of our methodology.

We create another separation graph for the results obtained on the debiased dataset as shown in Figure 7. As you can see, the distributions for white people and black people are a lot closer together compared to those seen in Figure 4, so the model is making much more fair decisions. Impressively, this is at a very small cost in accuracy of just 1.326% in accuracy.

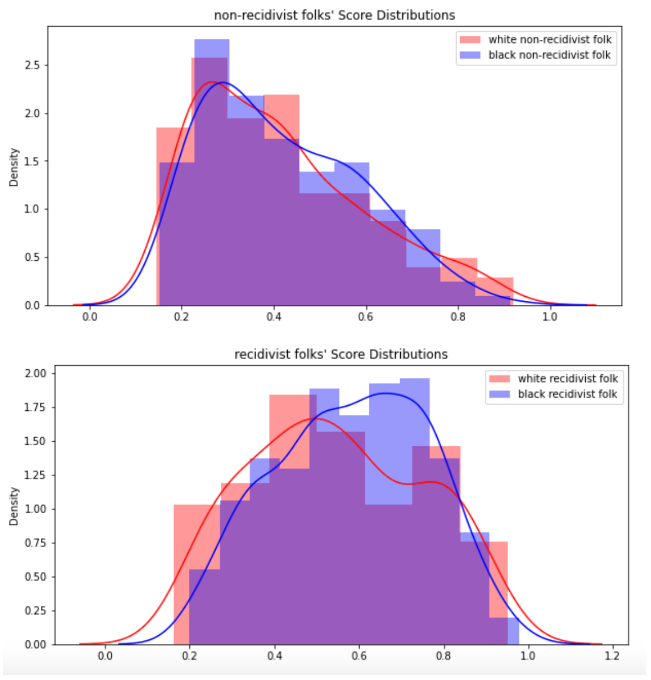

Fig. 7. Target Model on Debiased Dataset Separation Graph

6. *Retraining*

The same methodology of debiasing is further applied to the training data to create a medianed train dataset. Thus, a new debiased training dataset to predict the recidivism scores and a new model having the same architecture as the primary neural network is retrained without dependence on the paradigms of the existing weights. We run the model on the medianed test dataset and evaluate the results of accuracy and fairness as compared to the Target Model which was trained using the original dataset. Again, the distributions seen in Figure 8 are much more fair at only a small cost of 1.326%. The results of the evaluations are shown in Table 1 and 2.

Model	**Accuracy**
Original Target Model	67.614%
Target Model on Debiased Data	66.288%
Retrained Target Model on Debiased Data	66.288%

Table 1: Final accuracy measurements across different experimental models

Model	**TP-Parity**	**FP-Parity**
Original Target Model	34.648%	23.308%
Target Model on Debiased Data	10.955%	4.135%
Retrained Target Model on Debiased Data	10.569%	4.887%

Table 2: Final fairness measurements across different experimental models

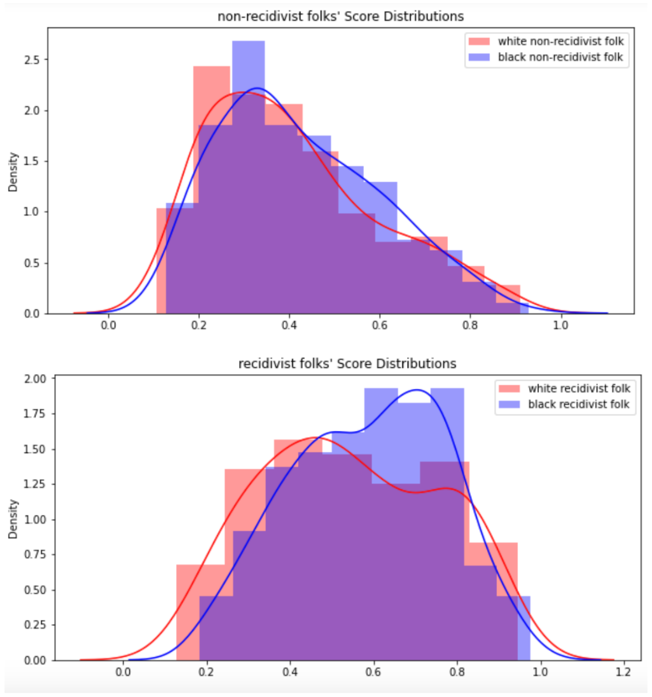

Fig. 8. Retrained Target Model on Debiased Dataset Separation Graph

We also wanted to ensure that debiasing the input data led to the Target Model generating hidden representations that did not encode the sensitive attribute (as much as they did original inputs). In other words, they should not contain as much information about race via redundant encodings as they did with the original inputs. To do this, we compare the accuracy of our sensitive model on the original test data and the debiased test data. We also retrain the sensitive model on the debiased train data and evaluate it on the debiased test data.

Model	**Accuracy**
Original Sensitive Model	66.004%
Sensitive Model on Debiased Data	56.534%
Retrained Sensitive Model on Debiased Data	54.261%

Table 3: Final accuracy measurements for predicting race (sensitive attribute) across different experimental Sensitive Models

5. Discussion

Overall, the results show that our approach is effective in eliminating bias in the Target Model. While the original Target Model had very high TP and FP parities, both the Target Model evaluated on the debiased data and the Retrained Target Model have much lower TP/FP parities, while achieving similar accuracy to the original Target Model. The Retrained Target Model had either the same or slightly higher accuracy since its weights were perhaps able to better match the distribution of the debiased inputs, depending on run. We also implemented the geometric repair as defined by [6] and saw that it achieved similar results in both fairness and accuracy, leading us to believe that this dataset was simply easy to debias without compromising accuracy. Other datasets that [6] tried on showed a significant ($\sim$20%) decrease in accuracy when dataset repair was implemented. Other datasets should be used to further substantiate our approach. Some slight issues to note are that we can see that the feature importances given by global interpretation seem to be a little random, so maybe the weights are not the best they can be. Next, we can see that we were unable to get stellar accuracy even with the original target model, so our accuracy did not have much room to drop with the target model evaluated on debiased data and the retrained target model. Nevertheless, it was encouraging to see the stark drop in TP/FP parities (combined with a very small drop in accuracy) on our Debiased and Retrained models, and that the sensitive model was no longer able to accurately predict the sensitive attribute given the debiased data, even after retraining. With the original data, the sensitive model was able to correctly classify race 66.004% of the time, which is significantly greater than that of a random classifier. But with the debiased data, the sensitive model was only able to correctly classify race 56.534% of the time, which is essentially the accuracy of a random classifier, only slightly improved. Retraining the sensitive model interestingly hurt the accuracy even further, thus fortifying our claim of reduced bias in the model

6. Future Work

There are a few areas that we noted could help spur further development on our project:

- It is quite possible that bias could arise in hidden representations of the neural network [2], even if it has been removed from each input feature individually. For example, perhaps a combination of low values for feature 1, and high values for feature 2 is highly correlated with a sensitive attribute, even if the distributions for feature 1/feature 2 appear similar among all sensitive groups. This is known as discrimination via representation. So far, this paper has focused on mitigating sensitive attribute leakage, but only at the input level (alleviating discrimination via input).
- Knowing that Global Interpretation can be done on hidden layers as well, we can propose to iteratively debias the inputs to the hidden layers in the Target Model using the same weighted median value replacement algorithm. For example let's say on the i^{th} iteration, the i^{th} layer of the network has debiased inputs. Then, we can see if any bias arose in the outputs of the i^{th} layer (inputs to the $(i+1)^{th}$ layer), and apply weighted median value replacement to them.
- Other work deals suppresses certain hidden neurons by using global interpretation to find which neurons are highly correlated with the sensitive feature. The outputs of these neurons to zero to turn off the correlation between sensitive attribute and DNN model prediction [2]. We don't see the need to completely zero out these

values, when they could be debiased using weighted median value replacement.

- The Target Model can also be iteratively retrained, where if on the i^{th} iteration, since the first i layers have debiased inputs, the Target Model can be retrained to better match the debiased distribution of inputs to the i^{th} to perhaps attain some small accuracy benefits as we saw in this paper.
- Further research may not have to use global interpretation at all, and can perhaps resort to statistical measures such as the Pearson Correlation Coefficient. [3] used this method in Graph Neural Networks and adaptively clamped the weights of features that correlated with the sensitive feature.
- Local Interpretation methods can also be used to generate feature attributions instead of Global Interpretation as the feature attributions generated by Global Interpretation did not seem fully sound in our work. One way this could be done is using Local Interpretation to reason about specific misclassified examples (e.g. samples who were predicted by the sensitive model as "black" and wrongly predicted by the target model to "is_recid"), and aggregate this data over all such misclassified examples to see which features were being used to make these decisions. Then, these could be converted into weights for each feature to be used in the weighted median value replacement algorithm.
- Finally, testing our approach on a multitude of datasets will help demonstrate its versatility across multiple contexts and potentially illustrate any issues that arise when working with data from other scenarios.

7. Conclusion

In this paper, we looked at how fairness and bias in neural networks can be affected by the phenomenon of "sensitive attribute leakage", where non-sensitive input features become highly correlated with sensitive input features, causing the neural network to over-emphasize the importance of the sensitive attribute in the prediction of the final output. In order to prevent the leakage of sensitive attributes into non-sensitive features, we implemented weighted median value replacement, where the value of each neuron correlated with a sensitive attribute was replaced with a formula that took a weighted average of the original value and the median of the corresponding (same percentile) values across each sensitive attribute subgroup. By implementing weighted median value replacement using global interpretability methods and looking at fairness metrics to analyze the outcomes of implementing these changes, we show that we were able to significantly improve the fairness of the model without considerably sacrificing accuracy, which shows that our post-processing step can improve the performance of an already trained neural network, and that retraining can be done if necessary. In a world where neural networks become increasingly more powerful, easier to train, and widespread due to their generalizability, we hope that post-processing methods that require minimal customization per model such as weighted median value replacement can help reduce the effect of unfairness and bias in neural networks without sacrificing accuracy.

8. Acknowledgments

The authors would like to thank our professor, Dr. Risto Miikkulainen for his wonderful class lectures and guidance for our project, as well as our TA, Ishan Nigam, for providing help and feedback for our homework and project assignments. We have made our jupyter notebook and paper accessible at https://github.com/amitjoshi24/sensitive-attribute-containment-dnn-fairnessl

.

References

[1] Hardt, M., Price, E., Srebro, N.. (2016). Equality of Opportunity in Supervised Learning.

[2] Du, M., Yang, F., Zou, N., Hu, X. (2020). Fairness in deep learning: A computational perspective. IEEE Intelligent Systems, 36(4), 25-34.

[3] Yu Wang, Yuying Zhao, Yushun Dong, Huiyuan Chen, Jundong Li, Tyler Derr (2022). Improving Fairness in Graph Neural Networks via Mitigating Sensitive Attribute Leakage. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. ACM.

[4] Cortez, Valeria. "How to Define Fairness to Detect and Prevent Discriminatory Outcomes in Machine Learning." Medium, Towards Data Science, 23 Sept. 2019, https://towardsdatascience.com/how-to-define-fairness-to-detect-and-prevent-discriminatory-outcomes-in-machine-learning-ef23fd408ef2.

[5] Sundararajan, M., Taly, A., Yan, Q. (2017, July). Axiomatic attribution for deep networks. In International conference on machine learning (pp. 3319-3328). PMLR.

[6] Feldman, M., Friedler, S. A., Moeller, J., Scheidegger, C., Venkatasubramanian, S. (2015, August). Certifying and removing disparate impact. In proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining (pp. 259-268).

[7] Kamiran, F., Calders, T. (2012). Data preprocessing techniques for classification without discrimination. Knowledge and information systems, 33(1), 1-33.

[8] Slavin Ross, A., Hughes, M. C., Doshi-Velez, F. (2017). Right for the Right Reasons: Training Differentiable Models by Constraining their Explanations. arXiv e-prints, arXiv-1703.

[9] Du, M., Liu, N., Hu, X. (2019). Techniques for interpretable machine learning. Communications of the ACM, 63(1), 68-77.

[10] Jiang, R., Pacchiano, A., Stepleton, T., Jiang, H., Chiappa, S. (2020, August). Wasserstein fair classification. In Uncertainty in Artificial Intelligence (pp. 862-872). PMLR.