On The Viability Of Combating Bias Through Fairness-Aware Adversarial Ensemble Methods

A Deep Dive Into Synthetic Data Generation And Adversarial Debiasing

Dennis Vlasblom

27-06-2022

Contents

Introduction

Over the past few decades there has been a significant rise in the use of Machine Learning (ML) techniques in virtually every industry. ML has proven to be incredibly powerful and can be used for a variety of different applications. From personalizing product recommendations to early identification of tumours (Kourou et al. [2015]), it has been shown ML obtains the ability to determine things a regular human being would take years to do by hand. This is due to the fact most ML methods rely on analysing large amounts of historical data to, among other things, obtain predictions of future behaviour of a person or classifying a person as part of a certain group. While analyses based on such a wealth of data often result in highly accurate predictive models, the possibility still exists that they are prone to discrimination.

Consider the compelling case of COMPAS, a software based decision support tool widely used by US courts to determine whether a defendant is going to reoffend in the future. The tool has been found to incorrectly classify black defendants who did not reoffend as dangerous and white defendants who did reoffend as safe. This disproportionate classification of black defendants as having a high risk of reoffending, in contrast to white defendants is one of the main criticisms of the tool. However, it is still used in most courts (Julia Angwin and Kirchner [2016]). Another example of discriminatory misclassification is XING, which is a German job finding platform similar to LinkedIn. Research done by Lahoti et al. Lahoti et al. [2019] showed that when an employer wanted to see the ranked results for the job title Brand Strategist, men were ranked disproportionally higher than women with similar credentials. The models these companies created to determine their respective targets can exhibit this discrimination due to the same reason they are so effective, the large amounts of historical data. Discriminatory behaviour exhibited in the past is captured by the dataset, which is then learnt by models based on this data. This opens up the possibility the model will keep predicting based on this discriminatory data, reinforcing the discrimination further.

Amidst all of this, the EU will start enforcing new legislation regarding the use of Artificial Intelligence and by extension ML for decision making (Satariano [2021]). The need for less biased ML is therefore not only a moral but also becoming a legal issue, which every company will need to adhere to. Beyond legal repercussions, using biased ML can also be detrimental to a company's conduction of business. Due to the fact that most ML models are built on large sets of data, where the bias can creep in from an overrepresentation of certain groups, one can imagine the following. A company desires to market to those customers providing the highest value to the company, however, these customers only represent a small chunk of the total customer base. Training a marketing model on such data will most likely result in results pertaining to the overall customer base, rather than the desired group of high-value customers. It is in essence, biased against high-value customers, due to their lack of representation.

Another example of declining business quality is that of Uber and Lyft (Lu [2020]), where it was found that people using either service to go to and from non-white neighborhoods had to pay more than the other way around. This issue arose due to area codes being heavily linked to what type of people live there, making it a clear predictor for race, especially in the USA. Besides the obvious discrimination appearing in this case, having higher prices for your service in certain areas where it does not need to be, could make the people living there not want to use your service, especially if they find out through why their prices are so much higher.

Many methods have been developed to combat this unintentional discrimination, with varying approaches as to where in the analysis process the problem is tackled. Methods combating the problem at the pre-processing phase attempt to change the data input to be less discriminatory, making sure models using this data will not discriminate. In-processing methods attempt to solve the issue by changing the way models itself work by adding constraints to ensure fairness. Finally, post-processing methods aim to adjust the results of a regular model to be less biased.

While these methods are highly effective at reducing discrimination, some issues are still apparent. One that arises when accounting for fairness, is the loss of accuracy. In almost every case, applying fairness-aware methods will lead to a decrease in the accuracy of a model. Obviously this is an undesirable side effect, and gives rise to the question where the line has to be drawn. What loss of accuracy is acceptable for what increase in fairness? This research approaches these issues by using two adversarial ensemble methods, Generative Adversarial Networks and Adversarial Debiasing. Such methods aim to solve issues by adding an 'adversary' network on top of the main problem. The main network tries to solve the main problem as best as possible, while the adversary tries to solve its own problem as best as possible. Among other applications, an adversary could try to predict a protected variable or whether data is real or fake. The main network then gets updated based on the performance of both itself and the adversary.

The goal of this research is to find out which method(s) of reducing discrimination are of high enough quality that they can actually be applied by companies and governments to ensure fairness. Hence, the main research question of this paper will be: *'To what extent can novel fairness-aware adversarial ensemble methods reduce unintended discrimination?*

This main research question will be answered by answering a set of sub-questions first. These sub-questions are the following:

- (i) 'To what extent can fairness be improved by using Fair Generative Adversarial Networks?'
- (ii) 'To what extent can fairness be improved by using Adversarial Debiasing?'
- (iii) 'How does accounting for fairness impact the accuracy of predictive models?'
- (iv) 'How does a combination of both Fair Generative Adversarial Networks and Adversarial Debiasing perform?'

Literature

Disparate Treatment and Impact

Discrimination has been an important but delicate topic which has taken over most of the world over the past few decades, which is why it is important to clearly define what discrimination means in the context of this paper. In most literature regarding discrimination in data science, discrimination is split into two forms; disparate treatment and disparate impact. Disparate treatment can be defined as the act of intentionally discriminating against someone, whereas disparate impact can be seen as 'unintended discrimination' (Feldman et al. [2015]). An often used example of disparate impact would be someone's zip-code having a negative effect on obtaining a loan, due to the fact that a person's zip-code can be associated with race, thus unintentionally discriminating against their race. While disparate treatment can easily be combatted by enforcing anti-discrimination rules, disparate impact cannot. Therefore, when discrimination is mentioned in this research, disparate impact is what is meant, unless stated otherwise.

WAE vs WYSIWYG

Additionally, before analysing whether a result is fair or not, the term fairness itself has to be defined. The literature differentiates between two opposing types of fairness; 'We're All Equal' (WAE) and 'What You See Is What You Get' (WYSIWYG). An often used example to elaborate on the differences between these two views are SAT scores. The SAT is an college acceptance exam widely used across the US. According to the WYSIWYG view, SAT scores are a great determinant of whether someone will perform well at university, as it can clearly be linked to one's intelligence and capability. On the other side of the argument, the WAE view could see SAT scores as potentially having structural biases, such that a different distribution of scores among one group versus another should not mean a difference in capability. Therefore, the WAE view sees SAT scores as an unfair metric for accepting someone into a university, as everyone should have an equal opportunity when applying. Most fairness metrics sit in between both of these opposing views, with some more on the side of WAE and vice versa. The fairness metrics used in this research lean more towards WAE than WYSIWYG, as accuracy is a measure these metrics are fine with reducing if it leads to an increase in fairness.

Furthermore, there exists another split for fairness, specifically Group Fairness and Individual Fairness. Group Fairness can be achieved when two or more groups of different backgrounds are treated the same. Individual Fairness looks at it from a more local level, and wants to achieve fair treatment between individuals. One could get a result where group fairness is achieved, while individual fairness is not. This research will focus on several Group Fairness metrics to see how they affect discrimination in different contexts.

Fairness Metrics

There are several fairness metrics that will be used in this research. One of the most widely used in the industry is Demographic Parity. It measures group level fairness and assumes the outcome $\hat{y}$ is independent of the protected attribute s (i.e. race, gender). This asserts the probability of the outcome being 1, or the preferred outcome, given the protected variable is 1, or the privileged group should be the same as the probability of the outcome being 1 given the protected variable is 0, or the underprivileged group. The mathematical notation of Demographic Parity is as follows:

$$P(y = 1 |\, s = 1) = P(y = 1 |\, s = 0)$$

In this research, Demographic Parity is calculated with the Statistical Parity Difference. This ratio is the equivalent of subtracting the probability of a person from an unprivileged group being assigned the favourable label by the probability of a person from a privileged group being assigned the favourable label. In mathematical terms it equates to:

$$\text{Statistical Parity Difference} = P(y = 1 |\, s = 1) - P(y = 1 |\, s = 0)$$

However, as some research has noted, it is a somewhat flawed measure of fairness and can lead to a loss in utility (Hardt et al. [2016]). A better and often used fairness metric is the Equalized Odds criterion. This matches the true positive rate and the false positive rate for different values of the protected attribute s. For example, the probability of a qualified person to be hired and the probability of an unqualified person to not be hired should be the same across genders when using Equalized Odds. Given a binary predictor $\hat{Y}$, its mathematical notation is as follows:

$$P(\hat{Y} = 1 |\, Y = y, s = 1) = P(\hat{Y} = 1 |\, Y = y, s = 0),\ y \in \{0, 1\}$$

Further, Equal Opportunity is a milder version of Equalized Odds. It only needs the true positive rate to be equal among groups, i.e. only the probability of a qualified person being hired should be equal across genders. There is a case to be made for both measures, but Equalized Odds can sometimes lead to a lower accuracy level for the unprotected class than Equal Opportunity, due to the fact that it wants equality of the false positive rate (Hardt et al. [2016]). As such, its mathematical notation is a simpler version of Equalized Odds:

$$P(\hat{Y} = 1 |\, Y = 1, s = 1) = P(\hat{Y} = 1 |\, Y = 1, s = 0)$$

Equal Opportunity is measured by using the Equal Opportunity Difference metric, which measures the difference in recall scores (True Positive Rates) between the privileged and unprivileged groups.

Research Methodology

Neural Networks

The main method used in this research is built out of a multi-layered deep learning neural network. Neural networks are a collection of algorithms that aim to solve problems by working similarly to the human brain. Each network contains multiple layers, each containing at least one node and one or multiple connections with the nodes in the following layer. A node can be seen as a digital version of a neuron, as they occur in actual biological brains. It is connected to other nodes in the network and each node can send signals to other nodes. A node has an input and an output, with the output being equivalent to the input transformed by a certain (activation) function. An activation function can be as simple as a sigmoid function, transforming the input to a value between 0 and 1. There are many different types of activation functions, all differing in usefulness depending on the task to perform. The output then gets sent to the connected node(s), where its inputs will be transformed again to a new output. The connections between each node are called edges, which all have different weights marking the importance of each connection to the network. The final layer of the network produces an output, which can be, among other things, a prediction or a computer generated image.

The NN learns to improve its outputs through the use of backpropagation. Backpropagation consists of updating the weights and biases each epoch. An epoch is defined as one full run-through of the inputs through the entire NN. The first epoch starts out with base values for each weight and bias, with each new epoch adjusting them. This adjustment is based on the size of the NN output error, which often has to get minimized. Usually done through the use of Stochastic Gradient Descent, an optimization method, the NN adjusts the values for each weight between nodes until a certain amount of epochs is reached.

GAN Basics

GANs are a type of method introduced by Goodfellow et al. [2014] that utilizes two distinct neural networks. One neural network is set up as a 'generator' and another as the 'discriminator'. The goal of the discriminator is to determine whether a data input is real or fake, while the goal of the generator is to fool the discriminator into thinking the data it generates is real. This back and forth between both neural networks makes the generator highly effective at creating fake data. The generator starts with an input of latent noise, which is transformed through several layers in its neural network setup. The generator's output is of the same shape as the real data. The discriminator then gets either real data or fake data generated by the generator as input, and has to figure out if the data is fake or not. This leads to the optimization of a minmax game, where the discriminator aims to maximize its loss function:

$$\max_{D} \mathbb{E}_{x \sim P_{data}}[logD(x)] + \mathbb{E}_{z \sim P_z}[1 - log(D(G(z)))] \tag{1}$$

And the generator tries to minimize its loss function:

$$\min_G \mathbb{E}_{z\sim P_z}[1 - log(D(G(z)))] \tag{2}$$

Here $\mathbb{E}$ is the expected value, x is the actual data, z is the latent noise put into the generator $G()$, $D()$ is the discriminator, P is the data distribution either for the actual data or for the generated data and the log is just there for scaling. The discriminator wants to maximize the expected value of the predictions based on the actual data x while minimizing those based on the fake data $G(z)$. Because minimizing is the same as maximizing the opposite, it maximizes $\mathbb{E}_{z\sim P_z}[1 - log(D(G(z)))]$. This is also the reason why the generator tries to minimize $\mathbb{E}_{z\sim P_z}[1 - log(D(G(z)))]$, as it will lead to fake data that looks increasingly more real.

Wasserstein GANs

While basic GANs are great at performing the tasks they are meant to do, they are prone to several issues. One of these issues is mode collapse, which consists of the generator finding a way to generate data in such a way that the discriminator is fooled, while not actually generating data similar to the real data at all. Another big issue with basic GANs comes from the similarity measure used. In most basic GANs the Jensen-Shannon Divergence (JSD) is used to measure the similarity between the real and generated data, which can lead to vanishing gradients.

Solving these issues is the Wasserstein GAN (Arjovsky et al. [2017]). It does this by measuring the similarity based on the Wasserstein, or the earth-mover's, distance. Furthermore, it sets a 1-Lipschitz constraint on the discriminator, which can be seen as a constraint ensuring the norm of the gradient is at most equal to 1. This ensures gradients do not explode and was shown to give much better results. The base Wasserstein GAN has one issue though, which is the way it enforces the 1-Lipschitz constraint by weight clipping, which means setting the critic's weights within a compact space $[-c, c]$. Setting c to a large value can lead to long run times, as weights will take a while to reach their limit. Setting c to a small value could lead to vanishing gradients in certain situations.

This is solved by Gulrajani et al. [2017] by adding a gradient penalty to the loss function of the discriminator (critic from now on). This can be seen in the final part of equation (3). The gradient penalty enforces the 1-Lipschitz constraint by directly constraining the norm of the gradient; $||\nabla_{\bar{x}} C(\bar{x})||_2$. This penalty is set on the distribution $P_{\bar{x}}$, which is defined as a distribution uniformly sampled from straight lines between pairs of points sampled from P_{data} and P_z. This results in an interpolation of two points, combining both samples into one. This is done based on a random weight ϵ that is assigned to the real data point, where $1 - \epsilon$ is assigned to the fake data point. The interpolation of these two points is called $\bar{x}$. Having the gradient's norm of the critic's prediction on $\bar{x}$ be strictly equal to 1 enforces the 1-Lipschitz constraint.

This is done due to the fact that the best critic should have coupled points in P_{data} and P_z connected by straight lines with a gradient norm of 1 (Gulrajani et al. [2017]).

All this changes the loss functions of both the critic and the generator. The critic's loss function then becomes:

$$\min_C = \mathop{\mathbb{E}}_{\hat{x}\sim P_z}[C(\hat{x})] - \mathop{\mathbb{E}}_{x\sim P_{data}}[C(x)] + \lambda \mathop{\mathbb{E}}_{\bar{x}\sim P_{\bar{x}}}[(||\nabla_{\bar{x}} C(\bar{x})||_2 - 1)^2] \tag{3}$$

Where $C()$ is the critic, x is a sample from the real data distribution P_{data}, $\hat{x}$ is a sample from the generated data distribution P_z, $\bar{x}$ refers to the interpolated sample, λ denotes the penalty coefficient for the gradient penalty and $\nabla_{\bar{x}}$ denotes the gradient with respect to $\bar{x}$.
The generator's loss function simply turns into:

$$\min_G = - \mathop{\mathbb{E}}_{\hat{x} \sim P_z} [C(\hat{x})] \tag{4}$$

Fair GANs

Several researchers have shown it is possible to adapt such GAN models further, to include constraints for fair data generation. FairGAN was proposed by Xu et al. [2018] and adds a second critic with the goal to predict the protected attribute given a generated sample. The generator aims to fool this critic, which continues until the protected attribute cannot be predicted by a generated sample, thus ensuring a lack of disparate impact. While this provided the first evidence for a GANs usefulness in bias reduction, the one used in this paper is based off of a GAN introduced by Rajabi and Garibay [2021]. They propose a Wasserstein GAN with only one critic that ensures fairness based on demographic parity. It works in two phases, where the first phase runs for a set amount of epochs and ensures similarity in data distribution. Here the critic's and the generator's loss functions are as equations (3) and (4), respectively.
The second phase aims to ensure fairness, by changing the loss function of the generator to include a constraint for demographic parity. As such, the loss function will get updated in several ways. First, the $\hat{x}$ is changed to not just be equal to the entire dataset, but to just the unprotected attributes. A second and third input is added, with $\hat{y}$ being equal to the output to predict (with a value of 1 representing the desired outcome) and $\hat{s}$ being the protected attribute (with a value of 0 representing the unprivileged group). These inputs will additionally be used to represent a discrimination score based on demographic parity:

$$\min_G = - \mathop{\mathbb{E}}_{(\hat{x},\hat{y},\hat{s}) \sim P_z} [C(\hat{x}, \hat{y}, \hat{s})] \; - \lambda_f (\mathop{\mathbb{E}}_{(\hat{x},\hat{y},\hat{s}) \sim P_z} [\hat{y}|\hat{s} = 0] \; - \mathop{\mathbb{E}}_{(\hat{x},\hat{y},\hat{s}) \sim P_z} [\hat{y}|\hat{s} = 1]) \tag{5}$$

Where λ_f denotes the weight given to the fairness constraint. The part in brackets represents the discrimination score, which is, based on the definition of demographic parity, equal to the difference between $P(y = 1| \; s = 1)$ and $P(y = 1| \; s = 0)$. When the expected value of $P(y = 1| \; s = 1)$ is bigger than that of $P(y = 1| \; s = 0)$ (when the privileged group gets better results), the loss will be larger. The loss function aims to be as small as possible, which will only happen if the opposite is the case (when the unprivileged group gets better results). This is how the second phase ensures fairness based on statistical parity.

Before describing the actual structure of the network, some notations will follow to make it clearer to understand. A tabular dataset is built out of N_C numerical columns $\{c_1, ..., c_{N_C}\}$ and N_D categorical columns $\{d_1, ..., d_{N_D}\}$, which are then one-hot encoded. The numerical values are scaled by estimating the number of modes and fitting a Gaussian mixture model to each of these columns. In this model the numerical columns are transformed using quantile transformation as follows:

$$c'_i = \Phi^{-1}(F(c_i)) \tag{6}$$

Here c_i represents the ith numerical feature, F is equal to the cumulative distribution function (CDF) of the feature and ϕ is the CDF of a uniform distribution. This all results in a dataset containing the following structure:

$$\mathbf{r} = c'_1 \oplus ... \oplus c'_{N_C} \oplus d'_1 \oplus ... \oplus c'_{N_D} \tag{7}$$

$$l_i = dim(d'_i) \tag{8}$$

$$l_w = dim(r) \tag{9}$$

Here $\oplus$ represents vector concatenation, d_i stands for the ith categorical columns' one-hot vector, l_i is the dimension of this vector, and l_w is the dimension of $\mathbf{r}$.

The network structure of this GAN can be seen in figure 1. The generator has an input size equal to the amount columns of the original dataset after the categorical columns are one-hot encoded, and is built out of a fully-connected first layer with a ReLu activation function and a second layer with several different activation functions. This due to the fact numerical variables and categorical / discrete variables have to be handled differently. To create an output similar to the original dataset, the nodes intended to create categorical variables are transformed into one-hot vectors by using multiple fully connected layers with a gumbel softmax activation of $FC_{l_w \to l_i}$.

The gumbel-softmax activation, first introduced by Jang et al. [2016], is an activation layer that allows for easy backpropagation through latent categorical variables. It is based on the gumbel-max trick for sampling categoricals (Gumbel [1954]). This trick is defined as taking the probabilities for each category, taking the logs of each and adding some Gumbel noise G_i to these individual logs, after which an argmax function will be applied to this set of values to determine which category will be predicted. The Gumbel noise can be sampled by taking two logs of a uniform, as seen in equation 10.

$$G_i \sim -\log(-\log(\text{Uniform}(0, 1))) \tag{10}$$

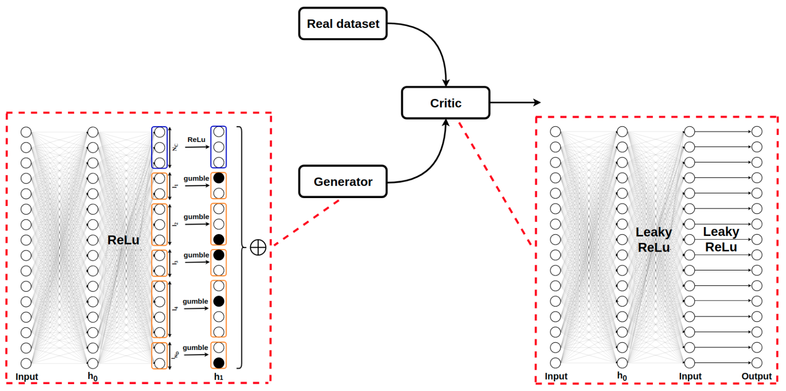
The gumbel-softmax layer used in this paper works in a similar fashion to the gumbel-max trick, but instead of applying the argmax function, it will apply the softmax function. The gradients from gumbel-max are zero, and as such nothing can be learned from them. The gumbel-softmax layer leads to continuous outcomes, such that gradients are not zero, which can be used to reach much better results for deep learning.

$$f_i(x) = \frac{\exp x_k/\tau}{\sum_{i=1}^{n} \exp x_i/\tau} \tag{11}$$

Equation 11 shows the softmax activation, where x represents the input, or the combination of the logs of the probabilities and the Gumbel noise. Additionally parameter τ represents a value between 0 and ∞ which controls how much of the 'winner-takes-all' dynamics occur when applying the softmax. When τ is very small, approximations get closer to the true categorical distribution, whereas when it gets bigger they get closer to a uniform distribution.

The numerical variables are simply transformed by a ReLu activation function of $FC_{l_w \to N_C}$. The results from these different vectors are then concatenated into one output layer.

Figure 1: GAN Network Setup



As can be seen in figure 1, the critic's network is simpler and just consists of two fully connected layers with Leaky ReLu activation functions. A formal representation of the generator's network is as follows:

$$\begin{cases} h_0 = Z \text{ (latent vector)} \\ h_1 = ReLu(FC_{l_w \to l_w(h_0)}) \\ h_2 = ReLu(FC_{l_w \to N_C(h_1)}) \oplus gumbel_{0.2}(FC_{l_w \to l_1(h_1)}) \oplus \\ gumbel_{0.2}(FC_{l_w \to l_2(h_1)}) \oplus ... \oplus gumbel_{0.2}(FC_{l_w \to l_{N_D}(h_1)}) \end{cases} \tag{12}$$

Here a fully connected layer with a and b as its input and output size respectively, is represented as $FC_{a \to b}$. $ReLu(x)$ and $gumbel_\tau(x)$ both denote the application of the respective activation functions on x, where τ denotes the parameter used by gumbel-softmax to control how closely to approximate one-hot vectors. The formal representation of the critic's network is as follows:

$$\begin{cases} h_0 = X \text{ (output of the generator or transformed real data)} \\ h_1 = LeakyReLu_{0.01}(FC_{l_w \to l_w(h_0)}) \\ h_2 = LeakyReLu_{0.01}(FC_{l_w \to l_w(h_1)}) \end{cases} \tag{13}$$

As before, $LeakyReLu_\tau(x)$ represents the application of the LeakyReLu activation function on x with slope τ.

Adversarial Debiasing

One of the most popular adversarial ensemble methods used for combating discrimination is the Adversarial Debiasing algorithm (Zhang et al. [2018]). It aims to remove the dependence of an output y on a protected variable Z, by creating a new adversary network g trying to predict Z given the predictor $\hat{y}$. The eventual goal is to create a model f that can best predict y given the inputs x, while at the same time making sure the adversary network is unable to predict the protected attribute Z. It works by first looking at the predictor and training it by modifying weights W to minimize its loss $L_{P(\hat{y},y)}$. The output layer of the predictor f is then used as input for the adversary network g attempting to predict Z, which will have weights U, a loss $L_{A(\hat{z},z)}$ and could have other inputs, depending on the definition of fairness it is trying to solve for. Each training step U is updated to minimize L_A with the gradient $\nabla_U L_A$, while W is updated with the following gradient:

$$\nabla_W L_P - proj_{\nabla_W L_A} \nabla_W L_P - \alpha \nabla_W L_A \tag{14}$$

Here α is a hyperparameter representing the adversary weight that can be modified each training step, $proj_{\nabla_W L_A} \nabla_W L_P$ makes sure the predictor does not move in a direction that helps the adversary, and $\alpha \nabla_W L_A$ tries to increase the adversary's loss. If both are not applied, the predictor will actually help the adversary and thus there would not be any debiasing. In the end this will result in the predictor to be rid of any dependence on the protected variable. To get better predictions, the learning rate for this algorithm is adjusted over time. When α is manually set, this is done by using a method known as the exponential learning rate decay. This adjusts the learning rate down by using the following mathematical formula

$$\text{Decayed learning rate} = lr_i * \text{dr}^{(\frac{gs}{ds})} \tag{15}$$

Here lr_i represents the initial learning rate, dr is a hyperparameter representing the desired Decay Rate, gs stands for the 'Global Step', or the current training step and lastly ds stands for the total amount of 'Decay Steps'. The Adversarial Debiasing algorithm used sets the dr to 0.96 and ds to 1000, or every 1000 steps the learning rate is decayed with a base 0.96. Additionally, $\frac{gs}{ds}$ is set to be an integer division, making the decayed learning rate follow a staircase function.

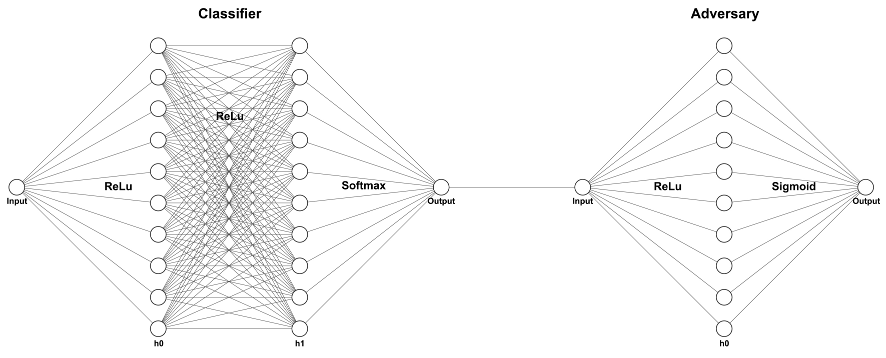
However, it was shown by Zhang et al. [2018] that in some cases, specifically when trying to enforce Equalized Odds, a different method can lead to better results. Instead of setting a fixed α, it can be increased over time by doing the following. α is set to $\sqrt{t}$, with t being the current step counter. Additionally, the learning rate is then adjusted using an inverse time decay instead of the exponential learning rate decay, as seen in the following formula.

$$\text{Decayed learning rate} = \frac{lr_i}{1 + \text{dr} * \lfloor(\frac{gs}{ds})\rfloor} \tag{16}$$

Here $\lfloor(\frac{gs}{ds})\rfloor$ represents the floor, or the greatest integer less than or equal to $\frac{gs}{ds}$, with the rest of the parameters being the same as above. In this case the dr is set to 0.1 and ds to 1000, or every 1000 steps the learning rate is decayed with a base 0.1.

This paper will look into the application of two different fairness constraints by use of Adversarial Debiasing, specifically Statistical Parity and Equalized Odds. This can be achieved by using different inputs for the adversary network g. For Statistical Parity, the adversary input is equal to the output of the predictor, which is to say the predicted label $\hat{Y}$. The adversary will try to predict Z based on this $\hat{Y}$. The gradient of g will then get included into the weight update rule of f, as to minimize the information g obtains about Z from $\hat{Y}$. Due to the fact that the Equalized Odds measure is based on True Positive and True Negative rates, solving for Equalized Odds requires an additional input. Therefore, both the output of the predictor ($\hat{Y}$) and the true label Y are used as inputs for the adversary g. Similar to before, this will limit the information g gets about Z. However, this time the adversary will learn the relation between Z and the true value Y regardless of what the predictor does, as it is simply given to the adversary. When the prediction $\hat{Y}$ gets given to the adversary, it will be able to improve its loss by learning the information included in $\hat{Y}$ that goes beyond what is included in Y. The predictor will therefore try to fool the adversary, by making sure $\hat{Y}$ does not include further information, making the model move towards Equalized Odds. As the widely used IBM implementation of Adversarial Debiasing does not account for different fairness constraints, an adjusted version of this algorithm created by Ball-Burack et al. [2021] was used for this, with some slight further adjustments.

Figure 2: Adversarial Debiasing Network Setup



The final network setup for Adversarial Debiasing looks very similar to what is presented in figure 2. The basic classifier consists of two hidden layers, both containing the ReLu activation, and one final softmax output layer. The adversary model has one hidden ReLu layer and one sigmoid output layer. The amount of hidden units in each layer was set to 100, but was kept to 10 in the image to keep it relatively small.

Validation Methods

Before applying the above mentioned fairness metrics, first the quality of the generated data needs to be analysed. This is done, based on research done by Mottini et al. [2018], by running several non-parametric tests to see whether the generated data distributions are similar to their original counterparts. To properly run these tests, the Euclidian distance between a point in the generated dataset and one in the original train and test set is calculated. This results in two distributions, one based on the distance between the generated data and the train set, and one between the generated data and the test set. Due to the data having both continuous and categorical variables, the critic's output is used for comparing nearest-neighbours, as this represents all the samples in a numerical space.

The first test is the two-sample Kolmogorov-Smirnov (KS) test, which tests for differences in the shape of two distributions. Its null hypothesis can be defined as H_0: The two samples drawn are from populations with the same distribution. The second test is the Wilcoxon Mann-Whitney U test, which tests whether there is a difference between two groups, it determines whether the GAN data is closer to the train or the test set. Under its null hypothesis the distributions of two groups are seen as equally close. If these tests result in a p-value above 0.05, the null-hypotheses of these tests cannot be rejected, and will therefore show whether the distributions are similar. Should this be the case, it can then be assumed the GAN has learned to generate data with high similarity to the original distribution.

Further methods of validating data similarity used are simple distribution comparisons, looking at the distribution graphs of all variables and comparing the real data with the generated data to see if big differences occur. Similarly, such distribution comparisons conditioned on the outcome variable or the protected variable could give useful insights into how the GAN changes the data to result in more fair predictions. Lastly, a comparison of correlations will be performed, to determine whether the correlations of the original data are changed significantly. This particular dataset (described in **Data**) will include non-normal distributions, therefore the Spearman rank correlation will be used for continuous variables. Similarly, the dataset contains several categorical variables, which is why the Cramer's V metric will be used for determining connections between these variables. This metric is based on the Pearson's Chi-squared statistic, and was first revealed by Cramer [1946]. It has a range of 0 to +1, and will therefore only show the size of the connections between variables, not the sign (whether the connection is positive or negative).
Due to the fact that different categorical variables have different amounts of categories, simply looking at the size of the correlation is not enough. Therefore, several rules of thumb were created to determine the impact size of a specific correlation. This paper uses a rule of thumb first mentioned by Cohen [1988], which will determine whether the impact of a correlation is either large, medium, small or negligible. It is based on the degrees of freedom between two categories, defined as the smallest number of categories between both variables, minus one. When the degrees of freedom are equal to 1, the size of the correlations and their corresponding impact size (large, medium, small or negligible) are shown in the first row of table 1. When the degrees of freedom go beyond 1, the impact size cutoff point is determined by dividing the original cutoff point with the square root of the degrees of freedom. This rule of thumb can be extended up to any degrees of freedom.

df	negligible	small	medium	large
1	$0 < 0.10$	$0.10 < 0.30$	$0.30 < 0.50$	0.50 or more
2	$0 < 0.07$	$0.07 < 0.21$	$0.21 < 0.35$	0.35 or more
3	$0 < 0.06$	$0.06 < 0.17$	$0.17 < 0.29$	0.29 or more
4	$0 < 0.05$	$0.05 < 0.15$	$0.15 < 0.25$	0.25 or more
5	$0 < 0.04$	$0.04 < 0.13$	$0.13 < 0.22$	0.22 or more
.	.	.	.	.
9	$0 < 0.03$	$0.03 < 0.10$	$0.10 < 0.17$	0.17 or more

Table 1: Rule of thumb table

Data

The dataset this thesis looks at is the UCI Adult Census dataset. It is described as a census survey from 1996, with several demographic variables and a specific output variable, income, which is defined as a binary variable stating whether the respondent had a salary of over or under \$50.000 per year. While it is widely used for simple prediction analyses, it disproportionally predicts males to have a 'high' and females to have a 'low' salary. Table 2 shows the different variables included in the dataset, what type of variable they are and a short description of each.

Variable	Type	Range
age	Continuous	A respondent's age
workclass	Categorical	Whether respondent works for government, private or is unemployed
education	Categorical	Highest level of education achieved by respondent
education-num	Categorical	Same as education, only numeric
fnlwgt	Continuous	The estimated number of people each row represents
marital-status	Categorical	A respondent's marital status
occupation	Categorical	What line of work a respondent is in
relationship	Categorical	What type of relationship a respondent is in
race	Categorical	A respondent's race
sex	Binary	A respondent's sex
capital-gain	Continuous	A respondent's capital gain
capital-loss	Continuous	A respondent's capital loss
hours-per-week	Continuous	How many hours per week a respondent works
native-country	Categorical	What country a respondent it from originally
income	Binary	What income a respondent earns per year

Table 2: UCI Adult Census Dataset variable explanations

Some adjustments to the original data had to be made to make the GAN work optimally. Specifically, for some categorical variables, like *native-country*, the generated data would exclude categories with really low observations in the original data. The observations including these categories often comprised of less than one percent of the data, however no clear pattern was found as to when a category got ignored by the generator. A logical explanation for this is these categories had so little impact on the data, that the critic could be fooled by the generator if it did not include them. In an effort to make the data as close to the original as possible, several small categories were combined into one bigger category, as to not 'lose' this data. Additionally, when used for analyses, the variables *education-num* and *fnlwgt* were removed. The first was removed due to it being exactly the same as education, only in numeric form rather than text, and the latter was removed as it is a weight for the estimated number of people each row in the dataset represents in the actual world. These were therefore deemed unimportant for classifying whether someone earns either above or below an income of 50k. Furthermore, all of the categorical variables were one-hot encoded and the remaining continuous variables were min-max scaled to be between 0 and 1.

Results

FairGAN

Data Generation Quality

Initially, a GAN similar to the one described in **Fair GANs** was run for 230 epochs, with its lambda value set to zero, meaning no fairness constraint to adhere to. The loss graph for this GAN can be seen in figure 3, where the critic and generator loss reach their minimum value quite quickly, and steady out over time. To see how well this GAN is able to generate realistic data, a comparison of the real data distributions with generated distributions can be found in figures 4a-4d.

Figure 3: Loss graph for both the generator and the critic for the 'unfair' GAN

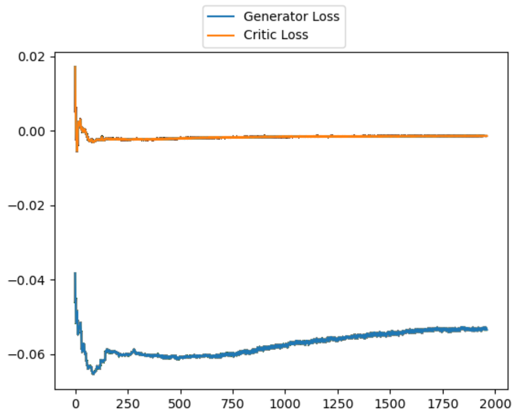
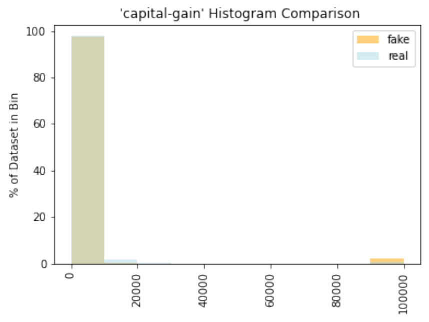
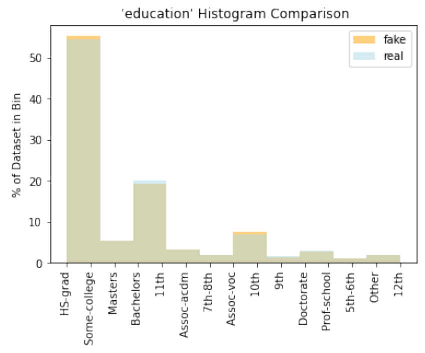


Figure 4: Real data distributions (light blue) vs Fake generated data distributions (orange)

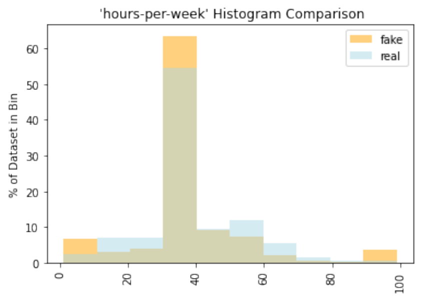
(a) capital-gain



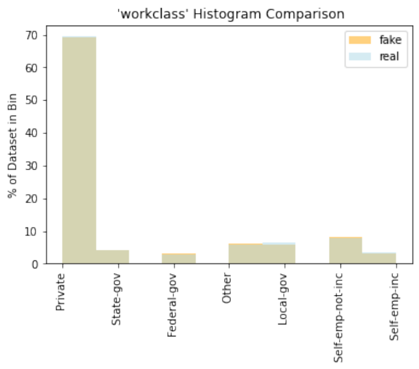
(b) education



(c) hours-per-week



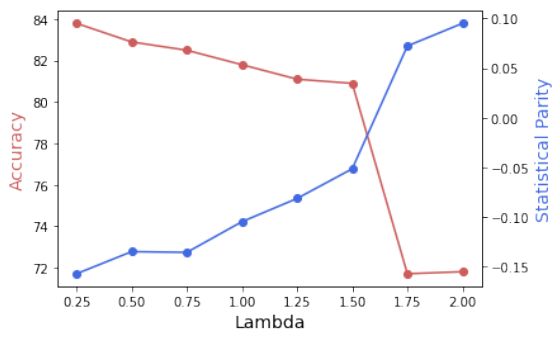
(d) workclass



As shown above, the basic generated data appears to be very similar to the original in terms of distribution, proving the GAN does its job well.

Moving on to the fair generated data, several values for lambda were tried, with the goal of achieving the lowest value for the Statistical Parity difference. These GANs with different lambdas were each run for 230 epochs, which include 60 'fair' epochs, where the loss function gets changed to include the fairness constraint. A simple classifier was then trained on the fair data and tested on the real data, to see how increasing lambda would impact the predictive accuracy and fairness, the results of which are shown in figure 5. A clear pattern can be seen, as accuracy decreases when lambda increases, whereas the Statistical Parity difference gets closer to zero when increasing lambda. The latter pattern breaks at a lambda value of 1.5, where the difference increases in the other direction, resulting in unfair predictions for males. Additionally, the accuracy suddenly drops by a lot. Therefore, the fair dataset mentioned in this paper used this value of 1.5 for lambda in its GAN fairness constraint, unless mentioned otherwise.

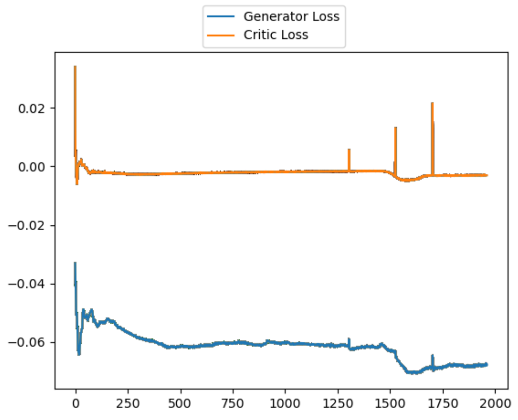
Figure 5: Effect of increasing lambda for the FairGAN *



* The Appendix shows a similar graph, replacing accuracy with likelihood

Figure 6 shows the loss graph for this fair data generator and critic, using a lambda value of 1.5. It was run for 230 epochs, of which in the last 60 epochs, the loss function gets the added fairness constraint mentioned in **Fair GANs**. As can be seen around that point, the loss drops quite a bit, before stabilizing again after some epochs. Running the GAN for more fair epochs eventually leads to the loss increasing quite a bit, which is why the cutoff point of 60 epochs was chosen.

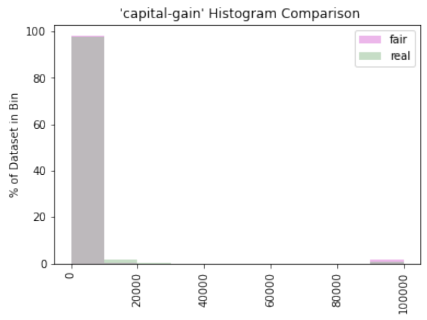
Figure 6: Loss graph for both the generator and the critic for the Fair GAN



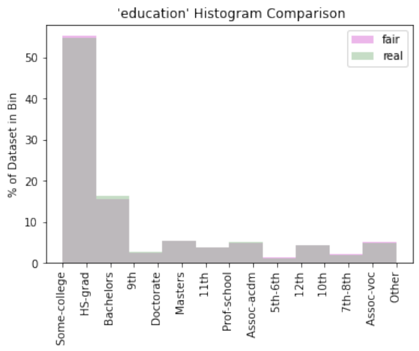
In the figures 7a-7d below, a similar distribution comparison as above can be found. They show the similarity between the real and the fair datasets. The only variable with notable differences, for both the fake and the fair generated dataset, is *age*. Figures 8a - 8b show this difference. It is unclear why this occurs, when other continuous variables do not have this issue. The Appendix includes all remaining images for distribution comparison.

Figure 7: Real data distributions (green) vs Fair generated data distributions (purple)

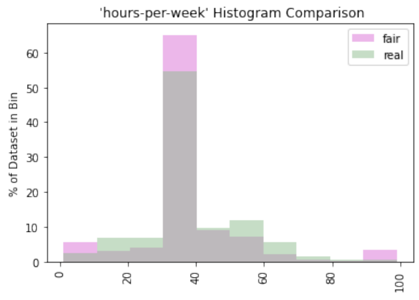
(a) capital-gain



(b) education



(c) hours-per-week



(d) workclass

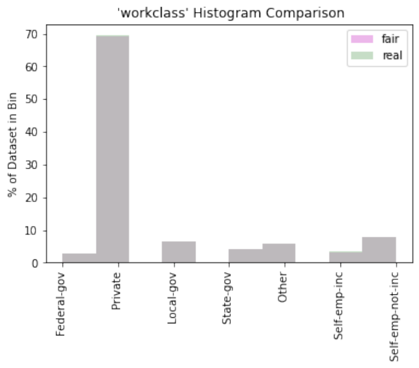
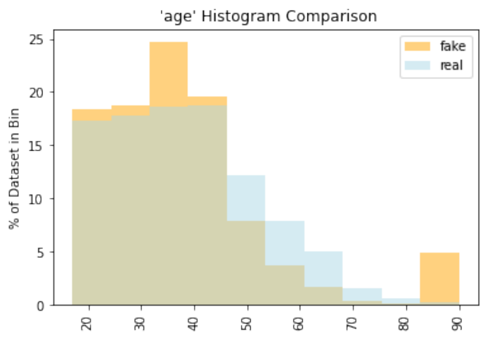


Figure 8: Real data distributions vs Generated data distributions for 'age'

(a) Fake



(b) Fair

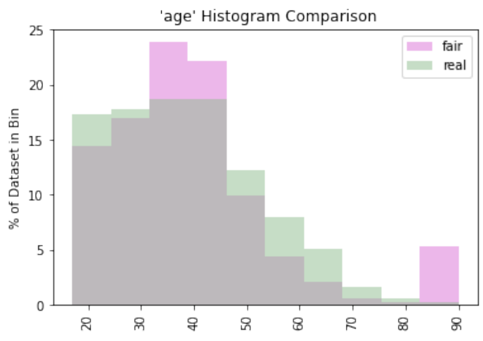


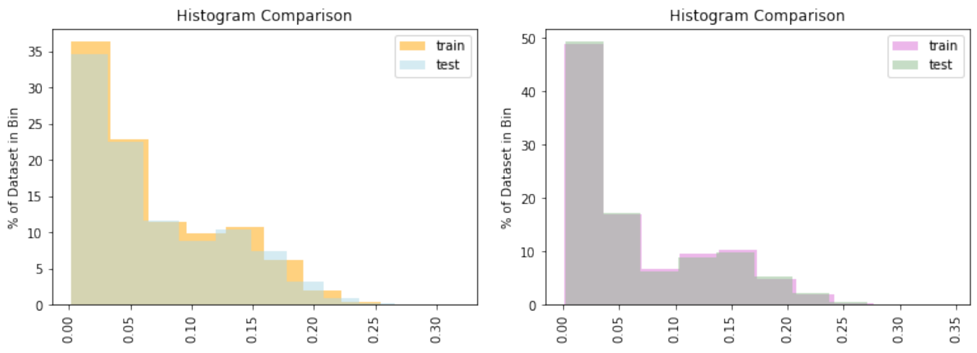
Table 3 shows the results for KS and Mann-Whitney U tests. The p-value for the KS tests in both cases is above 0.05, with the same being the case for the Mann-Whitney U tests. Furthermore, figures 9a and 9b show histograms for the distances between the generated samples and their nearest neighbours in both the training and test sets. Both show high similarities between the train and test sets. These results imply the GAN has learned to make distributions similar to the real data, rather than just copying it.

Test	Data	p-value
KS	Fake	0.707
KS	Fair	0.288
Mann-Whitney U	Fake	0.274
Mann-Whitney U	Fair	0.281

Table 3: Results for distribution comparison tests for both generated datasets

Figure 9: Distribution comparison of distances between GAN samples and their nearest neighbours in the training and test sets

(a) Fake (b) Fair



To see which variables influence each other most, tables 4 - 7 show the Spearman rank correlations for the continuous and the Cramer's V for the categorical variables. As mentioned in **Validation Methods**, each Cramer's V correlation has an impact size based on a rule of thumb. These impact sizes are shown in the tables next to the correlations. L stands for Large, M for Medium, S for Small and N for negligible.

variables	age	capital-gain	capital-loss	hours-per-week	income	sex
age	1.0					
capital-gain	0.124	1.0				
capital-loss	0.058	-0.066	1.0			
hours-per-week	0.147	0.092	0.06	1.0		
income	0.269	0.278	0.138	0.268	1.0	
sex	0.1	0.065	0.043	0.264	0.215	1.0

Table 4: Spearman correlation Table for original data

variables	age	capital-gain	capital-loss	hours-per-week	income	sex
age	1.0					
capital-gain	0.107	1.0				
capital-loss	0.016	-0.023	1.0			
hours-per-week	0.133	0.1	-0.018	1.0		
income	0.245	0.162	0.084	0.168	1.0	
sex	0.066	-0.008	-0.02	0.177	-0.206	1.0

Table 5: Spearman correlation Table for fair data

variables	workclass	education	marital-status	occupation	relationship	race	sex	native-country	income
workclass	1.0								
education	0.115 | S	1.0							
marital-status	0.093 | S	0.099 | S	1.0						
occupation	0.455 | L	0.2 | L	0.14 | M	1.0					
relationship	0.1 | S	0.122 | S	0.487 | L	0.174 | M	1.0				
race	0.058 | S	0.073 | S	0.083 | S	0.078 | S	0.098 | S	1.0			
sex	0.152 | S	0.093 | N	0.459 | M	0.41 | M	0.647 | L	0.114 | S	1.0		
native-country	0.039 | N	0.127 | M	0.062 | S	0.056 | S	0.072 | S	0.387 | L	0.054 | N	1.0	
income	0.182 | S	0.366 | M	0.448 | M	0.347 | M	0.454 | M	0.1 | N	0.215 | S	0.09 | N	1.0

Table 6: Cramer's V Table for real data

variables	workclass	education	marital-status	occupation	relationship	race	sex	native-country	income
workclass	1.0								
education	0.107 | S	1.0							
marital-status	0.1 | S	0.102 | S	1.0						
occupation	0.408 | L	0.212 | L	0.146 | M	1.0					
relationship	0.099 | S	0.118 | S	0.475 | L	0.175 | M	1.0				
race	0.058 | S	0.111 | S	0.084 | S	0.113 | S	0.102 | S	1.0			
sex	0.119 | S	0.124 | S	0.356 | M	0.393 | M	0.496 | M	0.098 | N	1.0		
native-country	0.054 | S	0.149 | L	0.072 | S	0.088 | M	0.08 | S	0.313 | L	0.109 | S	1.0	
income	0.165 | S	0.338 | M	0.382 | M	0.359 | M	0.413 | M	0.071 | N	0.206 | S	0.121 | S	1.0

Table 7: Cramer's V Table for fair data

As tables 4 and 5 show, most correlations seem fairly comparable between the original and the fair data. The most notable change for continuous values is to the *capital-gain* and *capital-loss* correlations with *sex*. They both shift from a positive to a negative correlation. Additionally, the correlation between *income* and *sex* has switched signs and become negative, a clear indicator of the GANs effect. When looking at the other correlations concerning the *income* variable, more differences occur. For all continuous variables except *age*, the correlations with *income* are lowered quite a bit. The most notable ones being the correlations between *capital-gain / capital-loss* and *income*. This is likely due to the GAN shifting several values for each sex, as will be shown later.

For the correlations between categorical variables, which can be seen in tables 6 and 7, most are of a small to negligent impact. Some large correlations occur, which are quite logical, like *race* and *native-country* being linked, as well as *sex* and *relationship*, as the latter comprises of categories including 'husband' and 'wife'. The variable *occupation* being largely connected to both *workclass* and *education* makes logical sense as well. It becomes interesting when comparing the real correlations with the fair ones. Specifically, the connection between *sex* and *education* changes from negligent to small. Curiously, in both the real and fair data, there is only a small connection between *income* and the protected variable *sex*. It is smaller in the fair data, but not by a lot. Overall these correlations look quite similar between the original and the fair data. Additionally, ensuring the correlations are not just some random approximation, the variance of each correlation was determined and looked at. This was done by splitting the sample in 10 equally large parts, then calculating the correlations for each pair of variables. The variance over these 10 correlations is then calculated, again for each pair of variables. These variances are then divided by 10 to determine the precision of the calculated overall correlation. The results for this can be found in tables 17 - 20 in the Appendix. For all correlations, the variance is very close to zero, which means the correlations are very precise.

Looking further into the differences the GAN makes when adjusting for fairness, figure 10 shows how many men and women are considered to be high or low income, in both the real and the fair data.

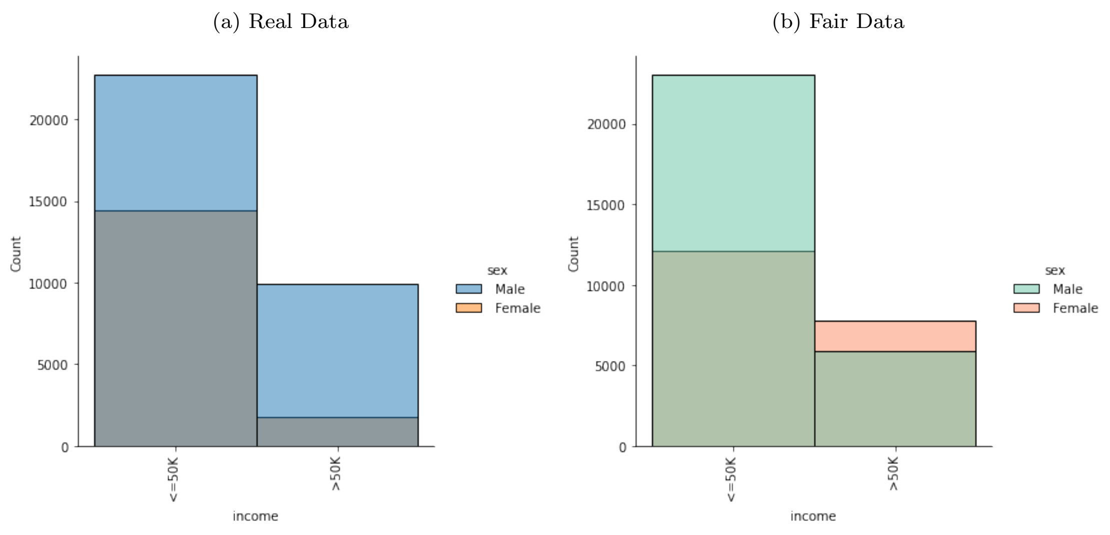


Figure 10: Comparison of 'income' distributions between males and females

(a) Real Data (b) Fair Data

As can be seen, the GAN opts to give relatively more women a higher salary. However, simply setting the income value for more women to high would make no sense, as most of these women are likely working jobs that pay relatively lower income. Therefore table 8 is included to show how many men and women work in each line of work, in both the real and the fair dataset.

Occupation	Real — Male	Fair — Male	Real — Female	Fair — Female
Craft-repair	5789	5441	323	789
Exec-managerial	4338	3152	1748	2698
Prof-specialty	3930	2856	2242	3166
Sales	3557	3230	1947	2451
Transport-moving	2228	2154	127	400
Other-service	2225	2256	2698	2829
Machine-op-inspct	2218	2228	804	940
Adm-clerical	1842	1597	3769	3899
Handlers-cleaners	1818	1671	254	109
?	1536	1474	1273	1341
Farming-fishing	1395	1344	95	91
Other	890	814	350	447
Tech-support	884	731	562	734

Table 8: Occupation comparison between real and fair data, conditioned on the protected variable

As can be seen, the GAN not only adjusts the income value of women, but also the occupation for women in such a way that more women are working higher paying jobs in the fair dataset. For reference, see table 9, which shows each occupation and what percentage of each income group is working in each field. As can be seen, most people earning a high salary are working in the *Exec-managerial* or *Prof-speciality* fields. These fields have more (less) female (male) representation in the fair dataset than in the real dataset, showing the behaviour of the GAN to not only put more women in the high income category, but also realistically changing their profession to correspond with this higher income.

Occupation	Real — >50K	Fair — >50K	Real — <=50K	Fair — <=50K
Exec-managerial	24.88%	22.56%	8.55%	7.84%
Prof-specialty	23.82%	23.10%	9.12%	8.12%
Sales	12.62%	13.09%	10.84%	11.06%
Craft-repair	11.83%	8.70%	12.73%	14.34%
Adm-clerical	6.57%	11.17%	13.04%	11.28%
Transport-moving	4.12%	3.19%	5.04%	6.03%
Tech-support	3.60%	3.90%	2.76%	2.65%
Machine-op-inspct	3.18%	3.10%	7.13%	7.81%
Other	2.70%	2.57%	2.49%	2.59%
?	2.27%	3.92%	6.85%	6.48%
Other-service	1.75%	4.07%	12.70%	12.89%
Farming-fishing	1.48%	0.56%	3.54%	3.87%
Handlers-cleaners	1.18%	0.07%	5.21%	5.04%

Table 9: Occupation comparison between real and fair data, conditioned on the 'income' variable

To further compare the real and fair data, table 10 shows the difference in sample means between each set. It shows a large increase in the sample mean for the *capital-gain* variable. What is notable, is the increase is larger for the female group than it is for the male group, further showing preferences. This increase also leads to men and women becoming more equal in terms of *capital-gain*, as in the initial situation men had double the mean capital gain of women, whereas the fair data makes it more or less equal between both groups.Similar behaviour of the GAN was seen above in tables 4 & 5, where the correlation between *capital-gain* and *sex* was largely reduced in the fair data as compared to the real data. Due to the fact that capital gain has become more equal across groups, it becomes less of a determining factor for *sex*.

Variables	Real — Male	Fair — Male	Real — Female	Fair — Female
age	39.494	38.208	36.928	37.440
capital-gain	1326.208	2696.475	580.726	2738.4022
capital-loss	100.410	51.734	61.475	70.842
hours-per-week	42.417	43.322	36.401	37.880

Table 10: Sample Mean comparison between real and fair data, conditioned on the protected variable

Base Classifiers

The basic classifiers, and by extension all classifiers following, were run for 200 epochs except when mentioned otherwise. The top row of table 11 shows the results when running a simple classifier on the original dataset. In terms of accuracy, it has a high level of competence, being 85.45% accurate in classifying which level of income a respondent has. However, as the table shows, the Statistical Parity Difference, is quite high at -0.165, meaning the probability for females to be classified as high earners is about 16.5% lower than the same probability for males. Similarly, the Equal Opportunity Difference sits at -0.078, showing the True Positive Rates for females to be lower than males by quite a margin as well. Both of these results lend credence to the classifier being unfair to women. However, the Equalized Odds Difference is 0.070, which means that when assessing fairness based on this metric, the original data favors women more than men. Running the exact same type of classifier on the fairness-constrained generated data shows the effectiveness of the GAN. As table 11 shows, the accuracy of the classifier drops by about 5%, while the Statistical Parity Difference now stands at 0.038, which means it has flipped to be more unfair to men, but is overall a lot more fair than the original classifier. The Equal Opportunity Difference in this case has decreased as well, to a value of -0.051, as well as the Equalized Odds Difference, which was reduced to 0.049. Additionally, an analysis was performed looking at how the base model trained on fair data would perform in classifying the real data. The results are shown on the third row of table 11 and show that the accuracy of the classifier has dropped to 80.92%. Further, the Statistical Parity Difference is down to 0.050, while the Equal Opportunity Difference has increased to 0.139, to the detriment of the male group. The Equalized Odds Difference has increased slightly as well. These results are far from optimal, as the reduction in predictive accuracy is pretty large.

Train Set	Test Set	Accuracy	Stat Par	Eq Opp	Eq Odds
Real	Real	**85.45%**	-0.165	-0.078	0.070
Fair	Fair	80.26%	**0.038**	**-0.051**	**0.049**
Fair	Real	80.92%	0.050	0.139	0.082

Table 11: Comparison of Basic Classifiers on both real and fair generated data

For all metrics, except accuracy, training and testing on fair data outperforms the rest. This is to be expected, as the generated data includes the added Statistical Parity fairness constraint. It is then also great to see that at least this fairness definition is satisfied when training on fair data and testing on real data. It shows the power of the GAN to create data that still predicts with quite a high level of accuracy while reducing biases based on specific fairness conditions.

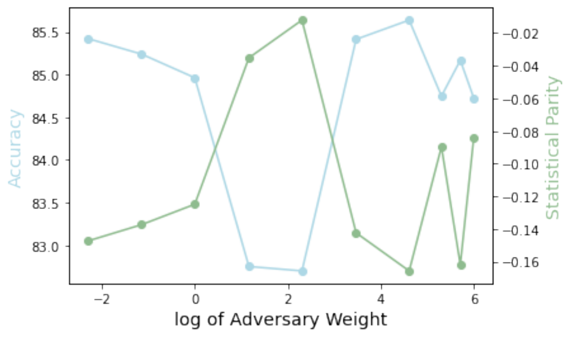
Adversarial Debiasing

Real Data

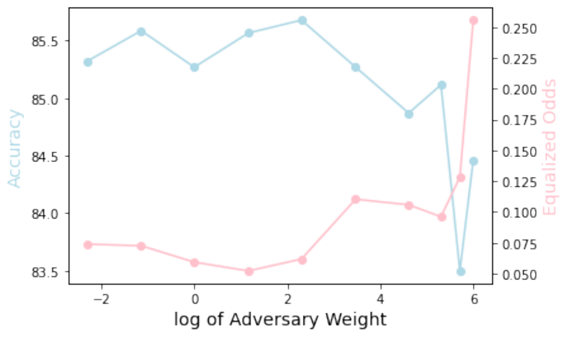
Similar to the comparison of different lambda values seen previously in figure 5, figures 11a - 11c show the effect of different adversary weights when applying the Adversarial Debiasing algorithm on real data. The Adversarial Debiasing algorithm used in this paper can be adjusted to account for either Statistical Parity or for Equalized Odds. Since the latter is similar to the Equal Opportunity fairness metric, a graph for that metric is included, to see if the algorithm proves useful for this metric as well. The weights that were tested for this algorithm are as follows, $\alpha = \{0.1, 0.32, 1, 3.2, 10, 32, 100, 200, 300, 400\}$. This is based on the weights used by Ball-Burack et al. [2021], with additional greater weights added, as in some cases more extreme values showed interesting results.

Figure 11: Comparison graphs between accuracy and fairness metrics for different adversary weights

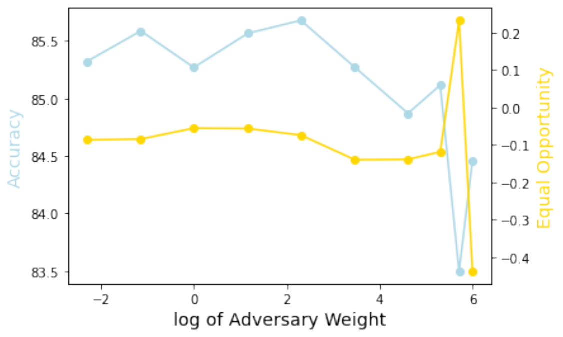
(a) Statistical Parity



(b) Equalized Odds



(c) Equal Opportunity



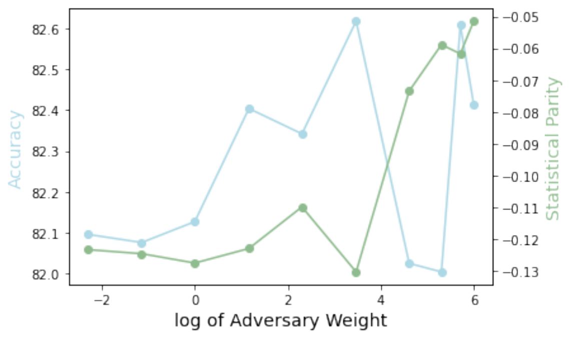
Important to note for the figures above, is that the optimal value for each fairness metric is zero. The metric is equal to the difference between each group from the protected variable, in this case male and female. Optimal fairness based on these metrics would mean a difference of zero. In some graphs that will be the top of the y-axis, in others the bottom. Furthermore, the log value of the tested weights is used as the x-axis, to make comparisons between lower weights more clearly visible. As can be seen, in the tested weight range, the optimal weight for Statistical Parity seems to be 10. As shown in table 12, this leads to a Statistical Parity difference of -0.012 and a reduction in accuracy from 85.40% to 82.70%. The optimal weight for Equalized Odds is 3.2, which has an Equalized Odds difference of 0.052 and a corresponding accuracy level of 85.57%. Finally, looking at how well accounting for Equalized Odds helps with Equal Opportunity, 1 seems to be the optimal weight, with an Equal Opportunity difference of -0.055 and an accuracy level of 85.27%.

Fair Data ($\alpha = 1.5$)

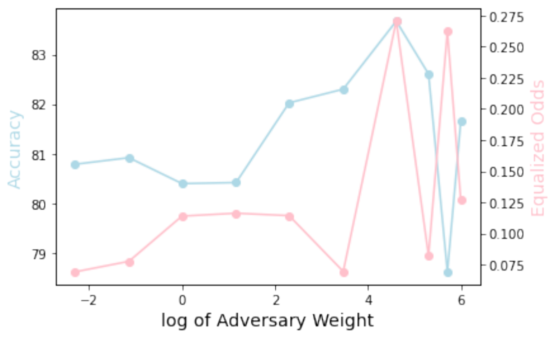
Beyond training the Adversarial Debiasing algorithm on real data, figures 12a - 12c additionally shows the effects of training it on the fair generated data, and testing it on real data.

Figure 12: Comparison graphs between accuracy and fairness metrics for different adversary weights, trained on fair and tested on real data

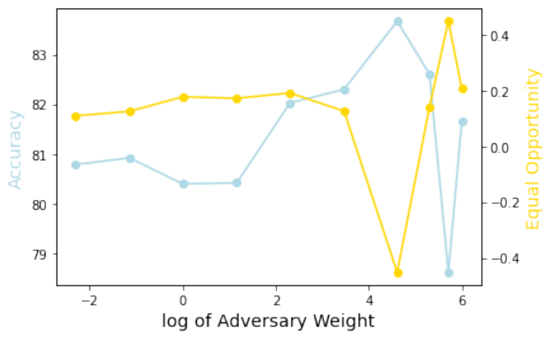
(a) Statistical Parity



(b) Equalized Odds



(c) Equal Opportunity



These graphs show again that lower weights seem to perform better for fairness, as heavily increasing the weight does not lead to better fairness results. However, for both Equalized Odds and Equal Opportunity, at a weight of 200, their fairness difference is similar to that of the lower weights, but has a much higher accuracy level. For Equalized Odds these values are a difference of 0.082 and an accuracy level of 82.61%. This trade-off is an important thing to consider when optimizing algorithms such as this; is the ultimate goal full negation of bias or is some slight bias allowed if it leads to better accuracy? For Equal Opportunity, the fairness difference is much worse than training and testing on real data, specifically at a weight of 200 the difference is 0.140 and the same accuracy level of 82.61% as Equalized Odds. For Statistical Parity, a different pattern seems to occur, where increasing the weight clearly helps with enforcing the fairness constraint. However, it does not outperform the original Adversarial Debiasing with weight 10, neither on accuracy nor on fairness. The final results for both real and fair training data can be found in table 12 below.

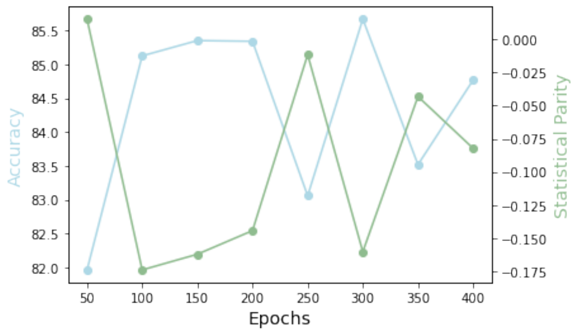
Model / Data	Train / Test	Weight	Accuracy	Stat Par	Eq Opp	Eq Odds
Base	Real / Real	0.0	85.45%	-0.165	-0.078	0.070
Parity Debias	Real / Real	10	82.70%	**-0.012**	0.252	0.135
Parity Debias	Fair / Real	200	82.00%	-0.059	0.118	0.069
EqOdds Debias	Real / Real	1	85.27%	-0.162	**-0.055**	0.059
EqOdds Debias	Real / Real	3.2	**85.57%**	-0.147	-0.056	**0.052**
EqOdds Debias	Fair / Real	200	82.61%	-0.041	0.140	0.082

Table 12: Comparison of Debiased Classifiers on both real and generated data

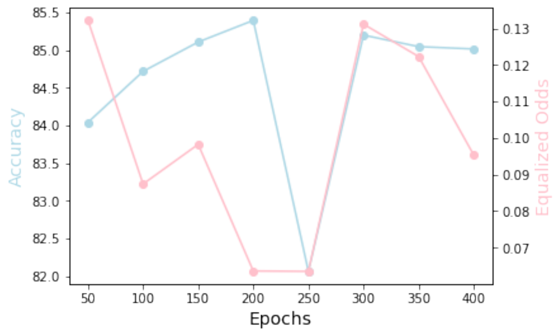
Additionally, as mentioned in **Adversarial Debiasing**, in some cases, specifically the Equalized Odds version of the algorithm, a continually updating weight could lead to better results. In the initial Adversarial Debiasing paper (Zhang et al. [2018]), this weight was set to $\sqrt{t}$, with t representing the current training step. This method was tested as well, to see how it compares to manual weight setting. Figures 13a - 13c show graphs similar to above, however now the x-axis does not represent the weight, but the amount of epochs the algorithm was ran.

Figure 13: Accuracy and fairness comparison with $\alpha = \sqrt{t}$, trained and tested on real data.

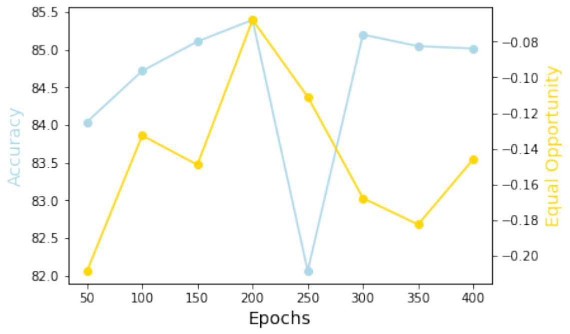
(a) Statistical Parity



(b) Equalized Odds



(c) Equal Opportunity



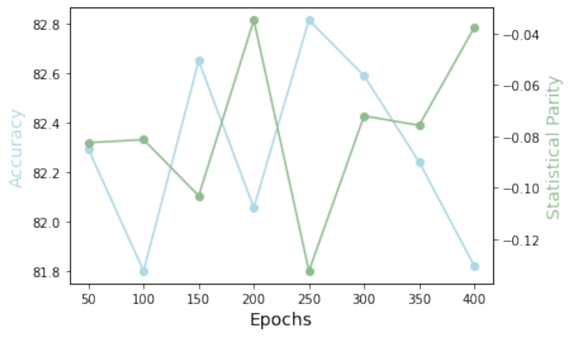
As can be seen, after 200 epochs, the fairness gained drops off quite a bit, and does not come back to its optimal level. This is especially true for Equalized Odds and Equal Opportunity. For Statistical Parity, the results curiously jump up and down. The original paper did suggest this variable weight only for the Equalized Odds version of the algorithm, and this might be the reason why. The optimal amount of epochs is hard to define, as one could run this algorithm for many more epochs, which could lead to better results. In any case, the best value found for Statistical Parity is 250 epochs, which leads to a difference of -0.012 and an accuracy of 83.06%. For Equalized Odds the best difference is 0.0635, which is obtained when running the algorithm for 250 epochs and leads to an accuracy level of 82.07%. However running for 200 epochs reaches a very similar difference, 0.0636, while having a much higher accuracy level of 85.39%. Equal Opportunity also reaches an optimal difference, that of -0.0678, at 200 epochs.

This was additionally trained on the data created by the FairGAN, for which the graphs can be found in figures 14a - 14c. It is clear that the combination of the fair data with Adversarial Debiasing can lead to better results for fairness, especially for Equalized Odds. Its difference lowers from 0.0636 to 0.0306, with accuracy dropping to 80.08%. For Equal Opportunity, the level of accuracy decreases from 85.27% to 80.08%, while its difference flips and slightly improves from -0.0678 to 0.0597. Again, this optimal difference for both Equalized Odds and Equal Opportunity is reached at 200 epochs, after which many additional epochs do not lead to better results. However, compared to training and testing on the real data, the reduction in fairness after 200 epochs is heavily reduced. For Statistical Parity, the results seem to jump all over the place, similar to before, however now they also lead to worse results. At 200 epochs, its difference changes from -0.012 to -0.035, with an accuracy drop from 83.06% to 82.06%.

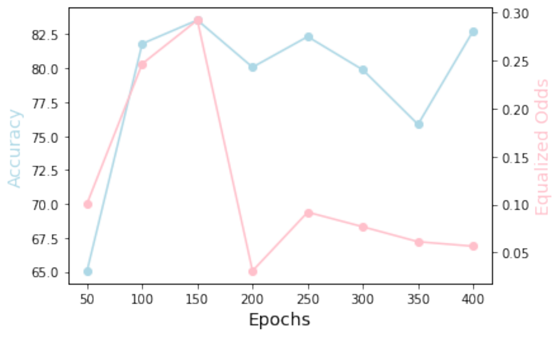
As can be seen in the appendix (images 21 - 22), the patterns that occur share a high similarity whether they were tested on either the (fair) training data, the (fair) testing data or the (real) testing data.

Figure 14: Accuracy and fairness comparison with $\alpha = \sqrt{t}$, trained on fair and tested on real data.

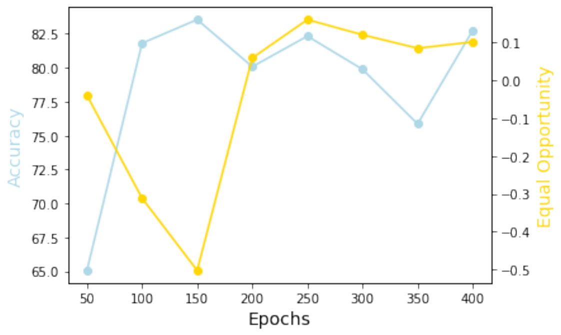
(a) Statistical Parity



(b) Equalized Odds



(c) Equal Opportunity



The optimal results of all this can be found in table 13. These results show that in some cases, the combination of Adversarial Debiasing with data generated by the FairGAN can lead to better results than without. Specifically, in the case where one does not want to manually set the optimal value for α, a combination of methods can lead to more fair results. Additionally, for Statistical Parity, the optimal value in both the manual and automatic α scenarios is the same, with the automatic scenario even leading to a higher accuracy level. This is not to say that this is the best value one can reach for Statistical Parity difference, but it does show that automatically changing α can lead to very good results, making it a viable and much easier alternative to manual weight setting.

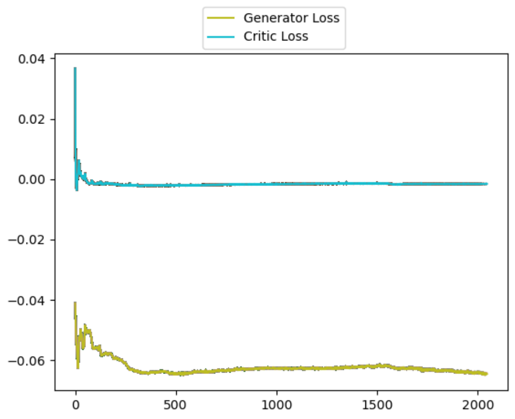
Model / Data	Train / Test	Epochs	Accuracy	Stat Par	Eq Opp	Eq Odds
Parity Debias	Real / Real	250	83.06%	**-0.012**	0.264	0.146
Parity Debias	Fair / Real	200	82.06%	-0.035	0.155	0.095
EqOdds Debias	Real / Real	200	85.39%	-0.160	-0.0678	0.0636
EqOdds Debias	Fair / Real	200	80.08%	-0.028	**0.0597**	**0.031**

Table 13: Comparison of Debiased Classifiers with $\alpha = \sqrt{t}$, on both real and generated data

Fair Data ($\alpha = 0.5$)

The previous results were all based on setting the FairGAN λ to 1.5. While these results were very decent, this research additionally looked into the effects of choosing a lower value or λ. The following results were all based on a λ of 0.5, and show interesting effects indeed. Starting with figure 15, showing the loss graph.

Figure 15: Loss graph for both the generator and the critic for the Fair GAN, $\lambda = 0.5$



The lower value for lambda also means a less extreme dip for the generator loss, with the loss remaining fairly similar to before the addition of the fairness constraint. Looking further into testing the new fair data, table 14 shows the results of running basic classifiers on this data. As can be seen, the accuracy level only drops by a bit, from 85.45% to 82.84% . This is less of a drop than shown with $\lambda = 1.5$ in 11, but clearly also leads to worse Statistical Parity, Equalized Odds and Equal Opportunity differences, some even worse than when training on fair data. Training on this new dataset and testing it on real data then unsurprisingly leads to underwhelming results.

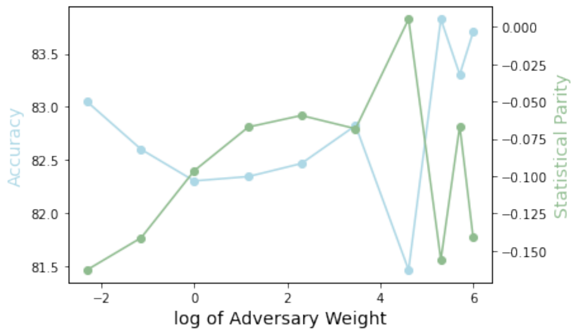
Train Set	Test Set	Accuracy	Stat Par	Eq Opp	Eq Odds
Real	Real	**85.45%**	-0.165	**-0.078**	**0.070**
Fair	Fair	82.84%	**-0.146**	-0.189	0.129
Fair	Real	82.98%	-0.169	-0.149	0.108

Table 14: Comparison of Basic Classifiers on both real and fair ($\lambda = 0.5$) generated data

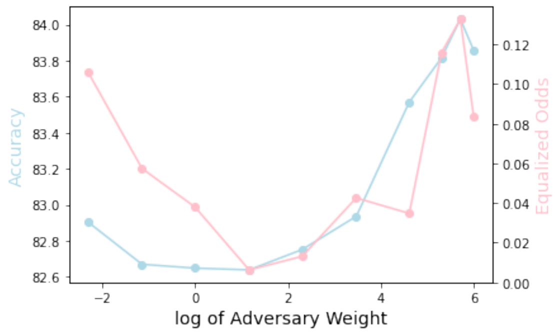
These results become interesting when looking at combining Adversarial Debiasing with the FairGAN data. As figures 16a - 16c show, the graphs for the different weights tested and what level of fairness and accuracy they correspond to.

Figure 16: Comparison graphs between accuracy and fairness metrics for different adversary weights, trained on fair ($\lambda = 0.5$) and tested on real data

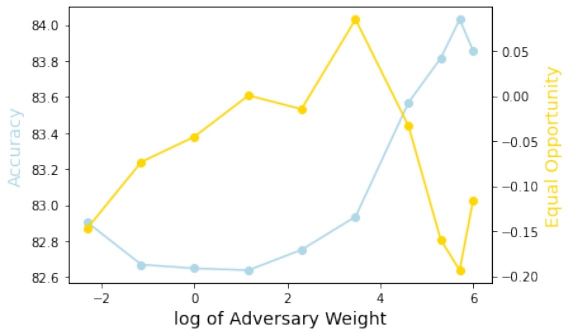
(a) Statistical Parity



(b) Equalized Odds



(c) Equal Opportunity



As is immediately noticeable, the Statistical Parity graph show the difference reaching almost a zero level. This difference of 0.005 was reached at a weight of 100, which is a great result, with the only caveat being the accuracy level, which sits at 81.46%. Moving on to Equalized Odds and Equal Opportunity, both their optimal differences can be found at a weight of 3.2, which leads to differences of 0.00624 and 0.00565 respectively, while having an accuracy level of 82.64%. Table 15 shows the optimal weights and their corresponding fairness values, with aforementioned results from table 12 included for comparison's sake.

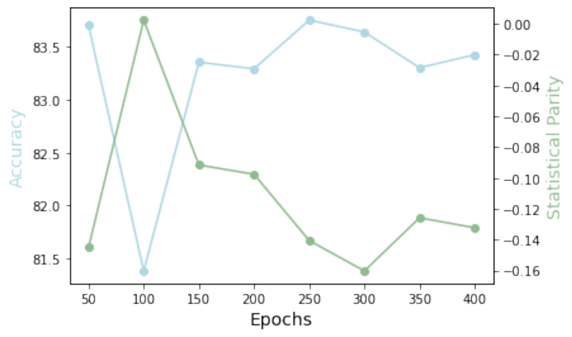
Model / Data	Train / Test	λ	Weight	Accuracy	Stat Par	Eq Opp	Eq Odds
Base	Real / Real	0	0.0	85.45%	-0.165	-0.078	0.070
Parity Debias	Real / Real	0	10	82.70%	-0.012	0.252	0.135
Parity Debias	Fair / Real	0.5	100	81.46%	**0.005**	0.247	0.147
Parity Debias	Fair / Real	1.5	200	82.00%	-0.059	0.118	0.069
EqOdds Debias	Real / Real	0	1	85.27%	-0.162	-0.055	0.059
EqOdds Debias	Real / Real	0	3.2	**85.57%**	-0.147	-0.056	0.052
EqOdds Debias	Fair / Real	0.5	3.2	82.64%	-0.098	**0.00565**	**0.00624**
EqOdds Debias	Fair / Real	1.5	200	82.61%	-0.041	0.140	0.082

Table 15: Comparison of Debiased Classifiers on both real and generated data

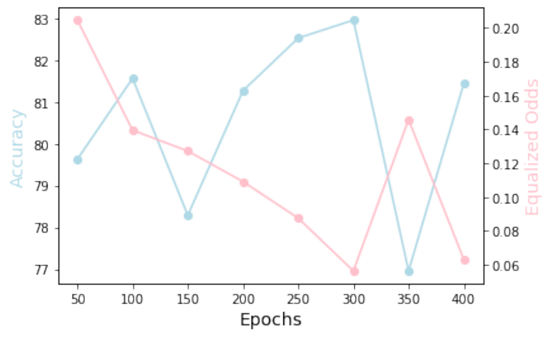
To see if similar results can be reached by automatically setting the adversary weight, figures 17a - 17c show what happens when setting $\alpha = \sqrt{t}$ with this data as a training set.

Figure 17: Accuracy and fairness comparison with $\alpha = \sqrt{t}$, trained on fair ($\lambda = 0.5$) and tested on real data.

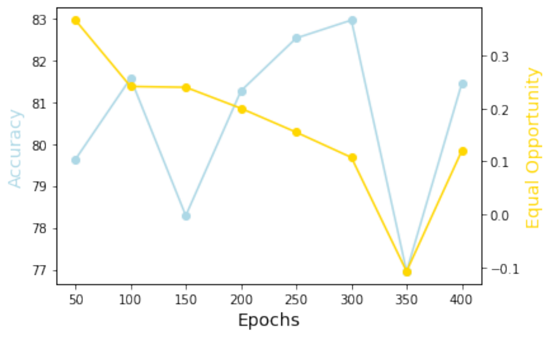
(a) Statistical Parity



(b) Equalized Odds



(c) Equal Opportunity



Statistical Parity reaches its optimal value very quickly, after 100 epochs already. This value sits at 0.002 and is therefore the closest this fairness metric has gotten to perfection, while its accuracy sits at 81.38%. While for both Equalized Odds and Equal Opportunity, these results are quite good, they do not outperform the $\lambda = 1.5$ dataset. The optimal value for Equalized Odds is 0.056 and is reached at 300 epochs, while its accuracy is 82.96%. For Equal Opportunity, the optimal value of -0.107 is reached a bit later, at 350 epochs, but is really close to its second best value, 0.108,which is reached at 300 epochs. They are both almost the same distance from zero, but the 'optimal' value carries with it a much lower level of accuracy. When going from 300 to 350 epochs, accuracy dips from 82.96% to 76.96%. Therefore, the 'optimal' value in this case is seen as the one reached at 300 epochs. As before, all of the results are included in table 16, with results from table 13 included for comparison.

Model / Data	Train / Test	λ	Epochs	Accuracy	Stat Par	Eq Opp	Eq Odds
Parity Debias	Real / Real	0	250	83.06%	-0.012	0.264	0.146
Parity Debias	Fair / Real	0.5	100	81.38%	**0.002**	0.250	0.140
Parity Debias	Fair / Real	1.5	200	82.06%	-0.035	0.155	0.095
EqOdds Debias	Real / Real	0	200	**85.39%**	-0.160	-0.0678	0.0636
EqOdds Debias	Fair / Real	0.5	300	82.96%	-0.132	0.108	0.056
EqOdds Debias	Fair / Real	1.5	200	80.08%	-0.028	**0.0597**	**0.031**

Table 16: Comparison of Debiased Classifiers with $\alpha = \sqrt{t}$, on both real and generated data

The methods used in this thesis are quite generalizable to other datasets. Both FairGAN and Adversarial Debiasing have relatively simple implementations, as to make the process of debiasing as easy as possible. However, one of the issues to look out for are the correctness of the generated data. One should check if all categories are included in the new data, and if not, one solution to tackle this has been provided in **Data**. Beyond this, the main difficulties with both methods come down to the hyperparameters. For the FairGAN, this is simply the λ value, for which different values work best for different datasets. As for Adversarial Debiasing, different sizes of hidden layers, the α value and the amount of epochs the model is run for should be tested, to see which set leads to the best performance. How many different options for each of these are used, is entirely dependent on what level of fairness vs accuracy is acceptable. This can be quite a time-consuming task, as the model has to be run multiple times, but at least for now there is no simple way of automating this process. Luckily, as shown in this research, there is no need to test hundreds of different values.

Conclusion

As this research has shown, a FairGAN can be a very useful tool for decreasing the biases that could be apparent in datasets. Its high quality of data generation combined with the reduction of biased results while still ensuring a high level of prediction accuracy makes it a very capable tool for ensuring fairness. When one tries training a classifier on fair data and then testing it on actual data, it shows good results as well. However, due to the inherent differences of the two datasets, the lambda value for the FairGAN often has to be increased to get to similar fairness results as when both training and testing on fair data. This will lead to a reduction in accuracy, which can be relatively large. Additionally, Adversarial Debiasing showed promising results as well, being able to create a greatly debiased classifier with very similar accuracy levels to the base classifier. Combining both methods can lead to better results, but are very dependent on the hyperparameters set by the researcher. For the GAN the pattern is quite clear, an increase in lambda leads to an increase in fairness at the cost of a reduction in accuracy, however for Adversarial Debiasing the story is quite different. Increasing the weight can lead to an increase or a reduction in fairness. Therefore it is quite difficult to know how large the weights to compare should be. One method showing a bit more consistency is the automatic weight setting, which sets α to a specific value each training step. This removes the need for manually determining each weight to see which one is optimal, and leaves it up to the amount of training steps to increase the weight. Overall this leads to better results in several cases. This does not work in all cases though, and thus further research into an automatic optimization of weights would prove incredibly useful. However, when done correctly, the outcomes can be really exciting, and lead to results that are very fair indeed.

This all shows that inherent biases contained in datasets can be significantly reduced, and that companies wishing to create and use more fair models definitely have the ability to do so. Whether these improvements are worth the reduction of model accuracy is up to the people actually applying these methods to determine. However, it is clearly not a one-size-fits-all scenario, and careful consideration of all parameters involved in the use of these methods is required if one wants to get the optimal results. Whether the application relates to racial or sexist bias reductions, or simply wanting to debias a model from the largest customer group, these methods are useful in any case. They are very flexible and can therefore be applied to multiple different scenarios.

One limitation to the current implementation of the FairGAN, is its lack of variety in fairness constraints. At the time of writing, the only publicly available version of a FairGAN can only account for Statistical Parity. This leads to better bias reduction for this fairness definition, but in some cases clearly negates the other definitions researched in this paper. A suggestion for further research would be to implement fairness constraints for Equalized Odds, Equal Opportunity and others. Additionally, this research tried to show what the FairGAN actually changes under the hood, trying to look into the black box that is a GAN by looking at the changes made to certain variables. However, this can only provide a base level of interpretation of the GAN, so another suggestion for further research would be to figure out how the GAN actually operates. Research performed by Härkönen et al. [2020] and Voynov and Babenko [2019] has looked into this before, however only in the context of image generation GANs. Similar methods could be applied to tabular data generation GANs, and could lead to very interesting results. Furthermore, while this research shows the power of Adversarial Debiasing, it only does so in the context of simple classifiers. The original paper mentions the ability of this algorithm to be applied to more complex classifiers, like Random Forests or Support Vector Machines. Further research could look into an application such as this.

Appendix

Figure 18: Effect of increasing lambda for the FairGAN

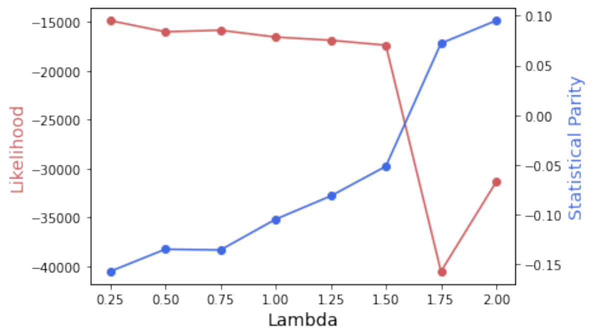
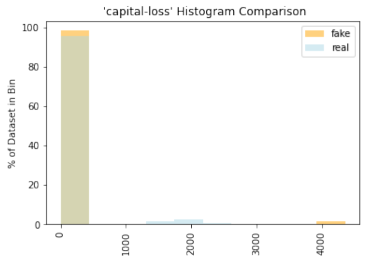
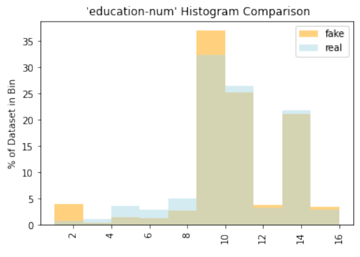


Figure 19: Real data distributions (light blue) vs Fair generated data distributions (orange)

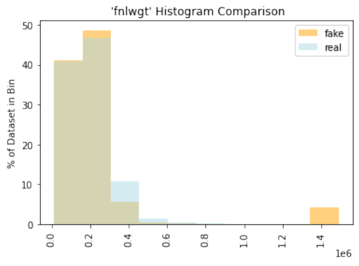
(a) capital-loss



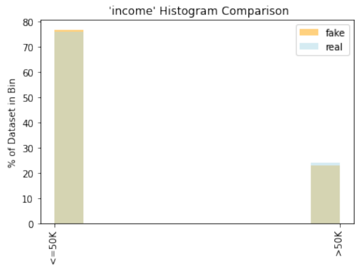
(b) education-num



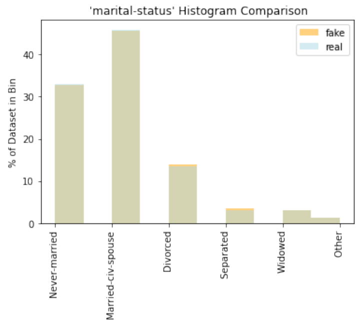
(c) fnlwgt



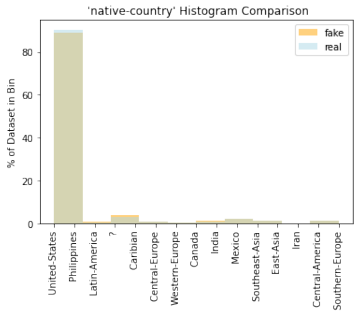
(d) income



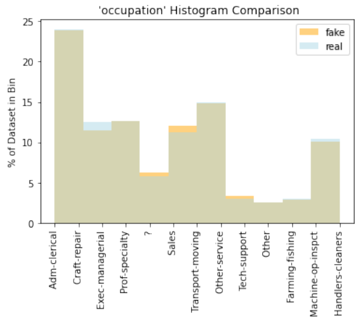
(e) marital-status



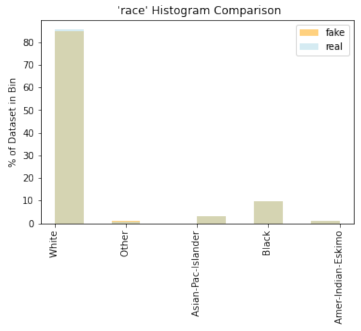
(f) native-country



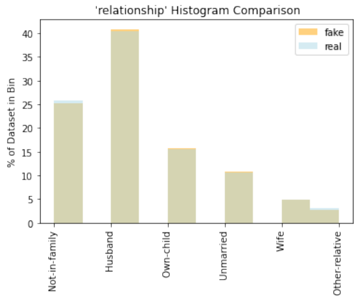
(g) occupation



(h) race



(i) relationship



(j) sex

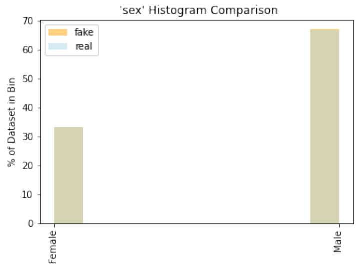
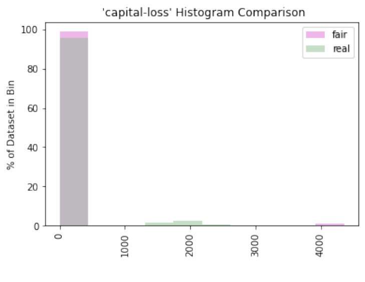
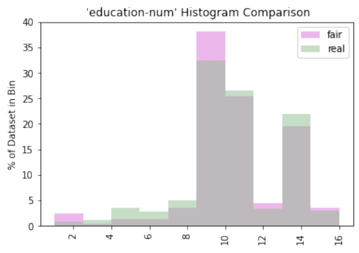


Figure 20: Real data distributions (green) vs Fair generated data distributions (purple)

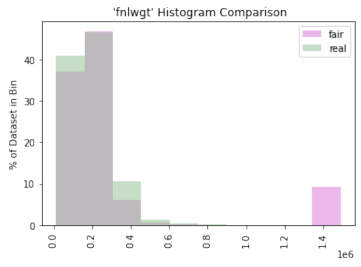
(a) capital-loss



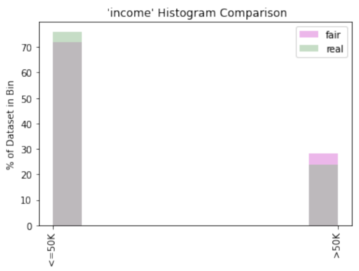
(b) education-num



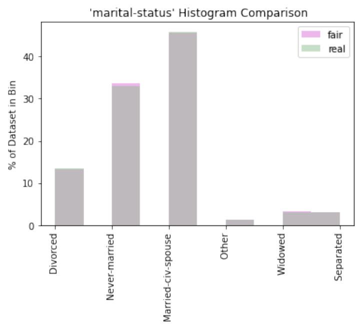
(c) fnlwgt



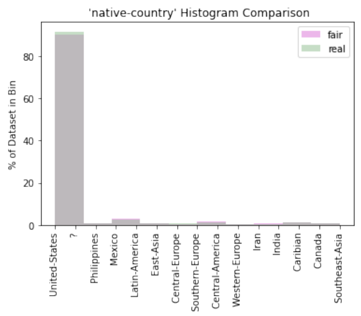
(d) income



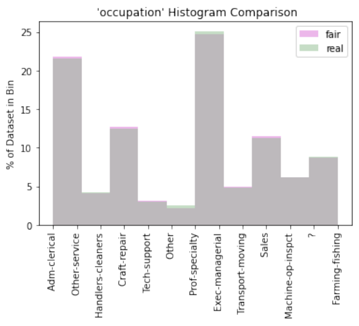
(e) marital-status



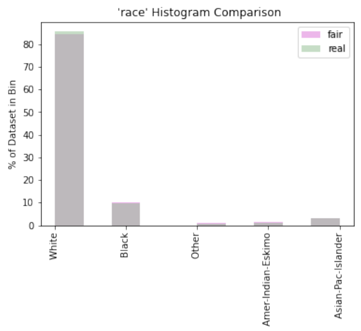
(f) native-country



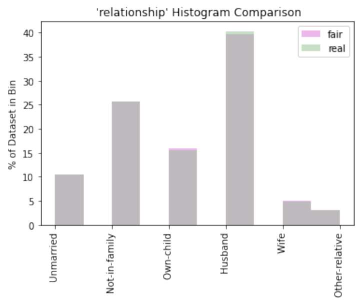
(g) occupation



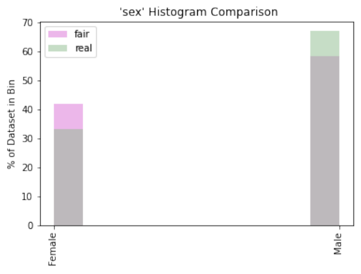
(h) race



(i) relationship



(j) sex



variables	age	capital-gain	capital-loss	hours-per-week	income	sex
age	0.0					
capital-gain	0.0002	0.0				
capital-loss	0.0001	4.803e-06	0.0			
hours-per-week	0.0004	0.0002	0.0001	0.0		
income	7.145e-05	6.456e-05	0.0003	0.0002	0.0	
sex	0.0002	0.0002	0.0001	0.0002	9.717e-05	0.0

Table 17: Spearman Variance Table for real data

variables	age	capital-gain	capital-loss	hours-per-week	income	sex
age	0.0					
capital-gain	0.0002	0.0				
capital-loss	0.0002	4.280e-06	0.0			
hours-per-week	0.0003	0.0002	0.0002	0.0		
income	0.0001	0.0007	0.0004	0.0002	0.0	
sex	0.0002	0.0001	0.0002	0.0001	0.0003	0.0

Table 18: Spearman Variance Table for fair data

variables	workclass	education	marital-status	occupation	relationship	race	sex	native-country	income
workclass	0.0								
education	9.461e-05	0.0							
marital-status	2.184e-05	1.511e-05	0.0						
occupation	8.593e-06	8.92e-06	2.324e-05	0.0					
relationship	5.563e-05	2.436e-05	1.450e-05	1.739e-05	0.0				
race	5.187e-05	4.812e-05	6.450e-05	3.406e-05	8.542e-05	0.0			
sex	7.233e-05	7.790e-05	0.0001	8.097e-05	3.859e-05	0.0002	0.0		
native-country	2.030e-05	6.288e-05	6.460e-05	9.066e-06	3.106e-05	0.0003	0.0002	0.0	
income	0.0002	0.0001	0.0001	2.407e-05	9.088e-05	3.8717e-05	9.717e-05	7.787e-05	0.0

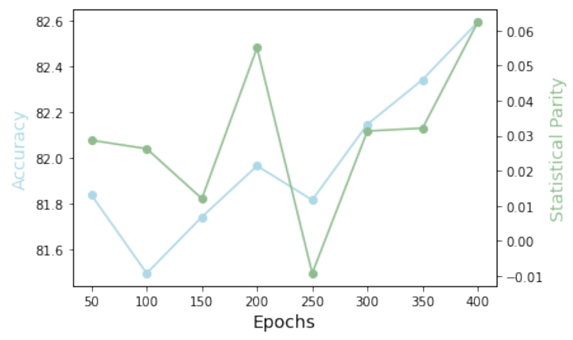
Table 19: Cramer's V Variance Table for real data

variables	workclass	education	marital-status	occupation	relationship	race	sex	native-country	income
workclass	0.0								
education	2.506e-05	0.0							
marital-status	5.974e-05	5.041e-05	0.0						
occupation	4.790e-05	5.368e-06	3.998e-05	0.0					
relationship	2.272e-05	5.887e-05	1.516e-05	4.152e-05	0.0				
race	1.876e-0	7.815e-05	0.0001	3.285e-05	0.0001	0.0			
sex	8.873e-05	0.0002	0.0002	6.042e-05	0.0003	0.0002	0.0		
native-country	2.603e-05	5.297e-05	8.191e-05	2.821e-05	2.677e-05	0.0002	7.639e-05	0.0	
income	0.0001	0.0001	0.0001	7.068e-05	0.0001	0.0001	0.0003	0.0001	0.0

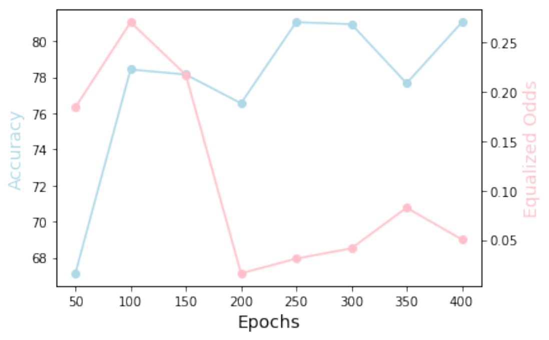
Table 20: Cramer's V Variance Table for fair data

Figure 21: Accuracy and fairness comparison with $\alpha = \sqrt{t}$, trained on and tested on fair training data ($\lambda = 1.5$).

(a) Statistical Parity



(b) Equalized Odds



(c) Equal Opportunity

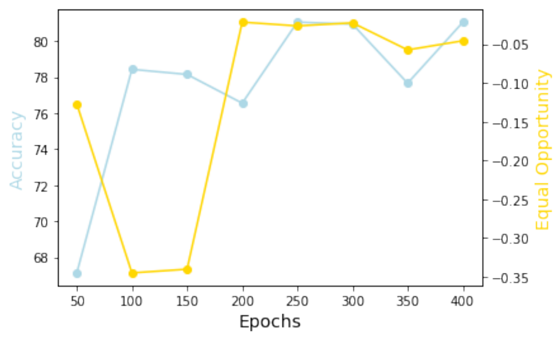
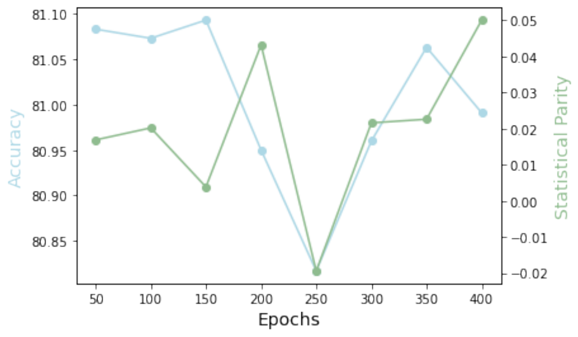
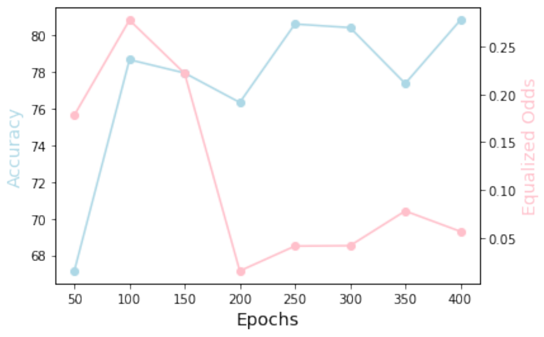


Figure 22: Accuracy and fairness comparison with $\alpha = \sqrt{t}$, trained on and tested on fair data ($\lambda = 1.5$).

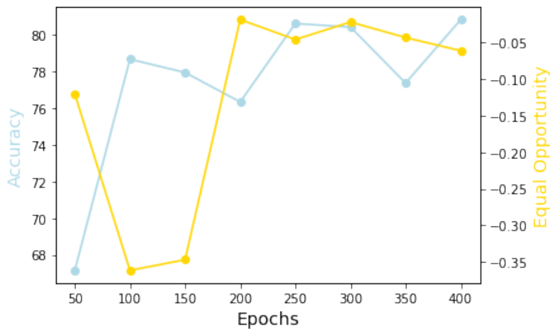
(a) Statistical Parity



(b) Equalized Odds



(c) Equal Opportunity



Bibliography

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein gan.

Ball-Burack, A., Lee, M. S. A., Cobbe, J., and Singh, J. (2021). Differential tweetment: Mitigating racial dialect bias in harmful tweet detection. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 116–128, New York, NY, USA. Association for Computing Machinery.

Cohen, J. (1988).

Cramer, H. (1946). *Mathematical methods of statistics / by Harald Cramer.* Princeton University Press Princeton.

Feldman, M., Friedler, S., Moeller, J., Scheidegger, C., and Venkatasubramanian, S. (2015). Certifying and removing disparate impact.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial networks.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of wasserstein gans. *CoRR*, abs/1704.00028.

Gumbel, E. J. (1954). *Statistical theory of extreme values and some practical applications: a series of lectures*, volume 33. US Government Printing Office.

Hardt, M., Price, E., and Srebro, N. (2016). Equality of opportunity in supervised learning.

Härkönen, E., Hertzmann, A., Lehtinen, J., and Paris, S. (2020). Ganspace: Discovering interpretable GAN controls. *CoRR*, abs/2004.02546.

Jang, E., Gu, S., and Poole, B. (2016). Categorical reparameterization with gumbel-softmax.

Julia Angwin, Jeff Larson, S. M. and Kirchner, L. (2016). Machine bias. there's software used across the country to predict future criminals. and it's biased against blacks.

Kourou, K., Exarchos, T. P., Exarchos, K. P., Karamouzis, M. V., and Fotiadis, D. I. (2015). Machine learning applications in cancer prognosis and prediction. *Computational and Structural Biotechnology Journal*, 13:8–17.

Lahoti, P., Gummadi, K. P., and Weikum, G. (2019). ifair: Learning individually fair data representations for algorithmic decision making.

Lu, D. (2020). Uber and lyft pricing algorithms charge more in non-white areas.

Mottini, A., Lheritier, A., and Acuna-Agost, R. (2018). Airline passenger name record generation using generative adversarial networks. *CoRR*, abs/1807.06657.

Rajabi, A. and Garibay, Ö. Ö. (2021). Tabfairgan: Fair tabular data generation with generative adversarial networks. *CoRR*, abs/2109.00666.

Satariano, A. (2021). Europe proposes strict rules for artificial intelligence. *The New York Times*, page 1.

Voynov, A. and Babenko, A. (2019). RPGAN: gans interpretability via random routing. *CoRR*, abs/1912.10920.

Xu, D., Yuan, S., Zhang, L., and Wu, X. (2018). Fairgan: Fairness-aware generative adversarial networks.

Zhang, B. H., Lemoine, B., and Mitchell, M. (2018). Mitigating unwanted biases with adversarial learning. *CoRR*, abs/1801.07593.