DEPARTMENT OF COMPUTER SCIENCE

Evaluating Combined Fairness Metrics in GANs

A novel way to create synthetic data for fairer binary classification

Elizabeth Hull

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Master of Science in the Faculty of Engineering.

Wednesday 21st April, 2021

Executive Summary

Bias in algorithmic decision-making can perpetuate and even exacerbate discrimination against disadvantaged subgroups of the population. Discrimination is also deeply embedded into the data we use to train algorithms. Various techniques have historically been employed to make this data and the models that use it more fair.

However, fairness itself is a complicated and contested topic. This work will explore the many ways in which fairness can be measured and demonstrate the theoretical impossibility of guaranteeing 'total' fairness in a binary classifier.

Next, drawing on adversarial training to generate debiased synthetic data, we combine two popular fairness metrics in the loss function of a GAN and generate synthetic datasets that show an overall improvement in these two metrics when used to train a binary classifier. The model can be weighted towards one metric or the other by changing the parameterisation of the loss function.

Finally, we will discuss the resulting fairness-accuracy trade-off and explore situations in which combined fairness might be helpful.

The main contributions of this project are:

- The first known GAN designed to make a trade-off between two fairness metrics
- An examination of how changing the weighting of this trade-off affects fairness
- A significant result that will aid multi-stakeholder decision-making where trade-offs between different fairness metrics are necessary

Acknowledgements

I am grateful to Rui Ponte Costa, Miranda Mowbray and Peter Bennett for all their help and support in this project.

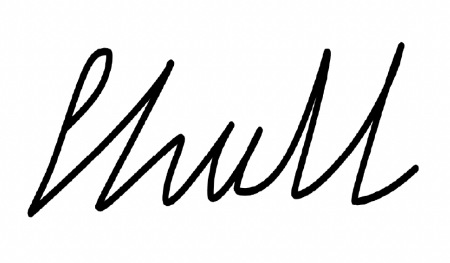
Thank you to Will Greedy and Theano Xirouchaki for their assistance in implementing the deep learning models.

Finally, I would not have been able to do this project without the love and support of my parents, Richard and Kathy. Thank you for always being there.

Declaration

This dissertation is submitted to the University of Bristol in accordance with the requirements of the degree of MSc in the Faculty of Engineering. It has not been submitted for any other degree or diploma of any examining body. Except where specifically acknowledged, it is all the work of the Author.

This project did not require ethical review, as determined by my supervisor Rui Ponte Costa.



Elizabeth Hull, Thursday 15^{th} April 2021

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Aims, Objectives and Deliverables

1.1.1 Aims

This study aims to investigate algorithmic bias and how to navigate the theoretical limitations of achieving total fairness. It will then design and evaluate a model that will enable complex decision-making to be fairer concerning protected attributes such as race, sex, or age. The model will be weighted between two different fairness metrics, helping to mitigate the lack of consensus on reducing algorithmic bias.

1.1.2 Objectives

This study will evaluate whether a synthetic dataset generated by a generative adversarial network (GAN) with a loss function optimised for a parameterised combination of demographic parity and equality of opportunity is fairer concerning the average of these two metrics than the average of two GANs with loss functions optimising demographic parity or equality of opportunity, without a significant loss of accuracy. If this trade-off is possible, it will critically examine how it achieves this, whether it has any significant negative side-effects and which types of real-world situations combined fairness could apply to.

1.1.3 Deliverables

- A comprehensive literature review justifying the importance of investigating algorithmic bias and evaluating the various approaches researchers have taken to design fairer algorithms.
- An exploration of why fairness metrics can be in conflict and why 'total' fairness is impossible to guarantee in all but the most trivial cases.
- Two GANs based on the DCGAN architecture and inspired by the Fairness GAN loss functions: one optimised for accuracy plus demographic parity and the other for accuracy plus equality of opportunity[27][30].
- A series of parameterised Combined Fairness GANs optimised for accuracy plus a weighted combination of demographic parity and equality of opportunity.

- An analysis of whether the novel GANs can achieve fairer results in a binary classification task than the average of the two GANs that incorporate only one fairness metric each.
- An evaluation of which policy settings or decision-making tasks these findings would be helpful for and why.
- Proof-of-concept for a system that would allow researchers to easily tailor fairness metrics to their particular problem space and context.

1.2 Motivation

In recent years, it has become increasingly common to use algorithms to make decisions that affect people's lives. *Allocative decision-making* is any process where finite resources have to be distributed according to eligibility criteria. Examples of allocative decisions that are now entirely or partly automated include loan approval, job hiring, college applications and even where to send police cars on patrol[6].

These algorithms take a feature vector X as input. For a loan application, the features might include the applicant's salary, credit score, assets and employment history. The algorithm processes these features and classifies an individual as loan-worthy or not. This paper will focus on similar cases where the outcome is a binary classification $\hat{Y} \in \{0, 1\}$, although it would be useful to extend a similar analysis to n-ary classification $\hat{Y} \in \{0, 1, ..., n\}$ in future work.

Many different types of algorithm can be used to supplement or replace human decision-making, from linear regression to state-of-the-art deep learning models. Automated decision-making algorithms have been responsible for 'enabling important advances' in many fields[6]. Many produce results more accurate than human judgement, particularly in cases where the relationship between the feature vector and the outcome is complex and multi-dimensional[7]. One commonly lauded advantage of algorithms is that, unlike their human equivalents, their decisions are not influenced by mood, energy or outside distractions[21]. They are also (usually) not guilty of *disparate treatment*: overtly allocating worse outcomes to societal subgroups by embedding unequal treatment directly in the algorithmic decision-making process.

Nevertheless, algorithmic decision-making can indirectly perpetuate societal biases towards protected subgroups (e.g. race, sex, age, disability status). For instance, past discrimination may be reflected in the training data, or the sampling of a minority group could be unrepresentative. When this is perpetuated or amplified by a machine learning model, it could lead to *disparate outcomes*. For example, a person might be denied a loan because previous data showed that people of the same socio-economic group default on their loans more often. However, this group may have suffered systemic economic hardships or discrimination that unfairly hampered their ability to pay back the loan.

In situations like this, our natural urge might be to redress the injustice and approve the loan. However, in a real-world decision-making situation, different stakeholders have different and sometimes conflicting needs. Shareholders will want to maximise the bank's profit. They may care less about potential bias in the data and place their trust in the person's estimated probability of loan repayment. In machine learning terms, they care most about the algorithm's accuracy. Activists may want the algorithm altered so that all socio-economic groups have an equal chance of getting a loan: a type of fairness called *demographic parity*. The applicant might argue that the system gave her a false negative: she is worthy of a loan even if it said she was not. She may think it would be fairer if all socio-economic groups had the same rate of false negatives: this is a different type of fairness called *equality of opportunity*. We will show that it is impossible to achieve all of these things at the same time except under very artificial constraints, so trade-offs have to be made.

Many papers have investigated the trade-off between accuracy and demographic parity or between accuracy and equality of opportunity. However, little has been done to examine if it is possible to make a tripartite trade-off between accuracy, demographic parity and equality of opportunity. If so, are there

real-world situations that could be improved by making such a trade-off? Are there meaningful compromises to be made in the arena of fairness?

1.3 Hypothesis

There exists a GAN that optimises for a weighted balance of demographic parity and equality of opportunity that is fairer with respect to these two metrics when its generated data is used to train a downstream binary classifier than the average of two GANs trained separately to optimise demographic parity and equality of opportunity. The 'combined fairness' GAN should also maintain a level of accuracy in the downstream classification task close to a classifier trained on real data.

The GAN will achieve this by using a composite loss function that is a weighted combination of the loss functions for the GANs that optimise demographic parity and equality of opportunity.

Note that *fairness* here is defined as the average of the demographic parity and equality of opportunity scores. We will see that these are just two of the many notions of algorithmic fairness. The proposed GAN does not constitute perfect or 'total' fairness but is instead a way of finding a compromise or trade-off between two contested fairness metrics.

A formal statement of the hypothesis can be found in the Methodology section. Demographic parity and equality of opportunity will be explained in more depth during the literature review.

1.4 Scope

Due to time limitations, this project will look solely at combining different fairness metrics in GANs. Many other generative models could potentially be used to generate unbiased data, such as variational autoencoders (VAEs) or hybrid VAE-GAN models. Proof of the hypothesis in GANs would not mean that either the general principle or the specific method would necessarily work in these other models.

Second, just two of many possible fairness metrics are investigated in this study (demographic parity and equality of opportunity). Investigating whether alternative metrics can be combined in a similar manner would be an ideal starting point for further research. It would be especially interesting to see whether equality of odds, a stricter extension of equality of opportunity, could also be weighted against demographic parity.

Third, the data set used comprises high-dimensional multimedia data (images with associated categorical feature vectors for subgroup and decision outcome). Again, this means that any results may not generalise to other kinds of data.

Finally, it may also be the case that a protected subgroup is more affected by the same amount of bias. For instance, Altman *et al.* argue that the 'negative effect of a criminal record on the likelihood of receiving a callback from a prospective employer is 40% greater' for black candidates than white candidates[3]. Furthermore, Roberts contends that the mass incarceration of black defendants also hurts the whole community by 'damaging social networks, distorting social norms and destroying social citizenship'[28]. These individual and social externalities will not be considered in this study.

Chapter 2

Literature Review

In recent years, the potential of algorithmic bias to harm protected subgroups has generated a great deal of attention in academia and the popular consciousness. A *protected subgroup* is defined as a subpopulation with characteristics protected by the 2010 Equality Act. These *protected characteristics* are age, disability, gender reassignment, marriage and civil partnership, pregnancy and maternity, race, religion or belief, sex and sexual orientation[1].

Although many other subgroups of the population experience discrimination, these subgroups' legal status highlights the seriousness of these particular kinds of discrimination and offers a clear real-world example of why stakeholders might be motivated to ensure algorithmic fairness beyond altruistic motives. Note that protected subgroups can be in the minority (e.g. disability, sexual orientation) but are not required to be (sex, marital status).

In this section, we look first at a specific example of algorithmic bias and its real-life consequences. We then turn to some of the different definitions of fairness that have been proposed as tools for measuring bias in machine learning algorithms. We critically examine the issues with these definitions and demonstrate why there is no one-size-fits-all way to implement perfect fairness. We then evaluate proposed solutions to this problem before finally looking at the application of fairness to high-dimensional multimedia data in GANs.

A note on the term 'bias'
It is important to clarify that the term *algorithmic bias*, as used in this paper, refers to perceived or actual unfairness within algorithmic decision-making. Readers may be familiar with a different use of the term in machine learning, which often arises when considering the trade-off between 'bias' and 'variance' in a model. In this second context, bias is related to the concept of underfitting, where a model does not have enough parameters to model the true distribution of the data correctly. While underfitting could potentially cause unfairness if the model underfits more on a particular subgroup of the data, it is not what is meant when discussing bias in this report.

The exact definition of which specific unfairnesses or inequalities constitute algorithmic bias is contentious and will be discussed later.

2.1 COMPAS: bias in action

One critical case of algorithmic bias that generated enormous academic and public interest was the COMPAS scandal of 2016. COMPAS, which stands for Correctional Offender Management Profiling for Alternative Sanctions, is an algorithm used by several jurisdictions in the United States to determine recidivism rates, i.e. the likelihood that a prisoner will re-offend after release. The prisoner is scored from 1 (low risk) to 10 (high risk), and judges use this score to help them decide whether the prisoner will be granted parole. The algorithm can be evaluated by following up with ex-convicts several years after release and comparing their actual recidivism rates with those predicted by the COMPAS algorithm[4].

A report by the non-profit organisation ProPublica found that although the COMPAS algorithm predicted recidivism rates for white and black defendants with similar accuracy, it was 'far more likely to assign a high risk score to black defendants who do not go on to re-offend'[6]. In statistical terms, black prisoners had a much higher false negative rate than white prisoners and bore the cost through unnecessary incarceration. Furthermore, white prisoners had a higher false positive rate.[1] These were cases where the algorithm evaluated prisoners as low risk, but they did commit another crime upon release[4]. High-risk white prisoners were thus more likely to benefit from early release than high-risk black prisoners.

Notably, the COMPAS algorithm does not explicitly use race as a feature in its model. However, even when a protected attribute is not directly included in a model, other attributes can act as proxies[7]. For example, it has been shown that black people are more often charged for minor drug offences than white people[32]. Prior criminal history is a feature in COMPAS, demonstrating that algorithms can 'carry forward earlier unjust treatment' by various 'social institutions that may foster disadvantage' to a particular subgroup[7]. Moreover, algorithmic modelling has the potential to augment as well as perpetuate existing discrimination[6].

Northpoint, the company that developed COMPAS, responded to ProPublica's criticism by citing the equal accuracy rates for black and white prisoners[15]. Although this rebuttal was widely condemned for not addressing the false negative and false positive rate disparities, it does exemplify how different stakeholders might weight different fairness metrics differently in importance.

2.2 What is fairness?

So what does it mean to say that a decision-making algorithm is fair? From an informal standpoint, it might seem straightforward: an algorithm is fair if it treats everyone the same. However, as soon as we try to define this notion formally, we run into complicated questions that span from the technical to the legal and philosophical.

Given our world has finite resources, we cannot give everyone what they want at all times. We might try to enforce fairness by making the 'fairest' decision and applying this to everyone. For COMPAS, we could either grant parole to everyone or deny parole to everyone. However, neither solution is realistic. Dangerous prisoners granted parole might commit further crimes that they would not have had the opportunity to otherwise. On the other hand, keeping all prisoners locked up forever would violate their human rights and displease the tax-paying public.

A different approach would be to set a quota for the percentage of prisoners allowed to be released on parole. Under this scheme, we can make arbitrary decisions about who gets parole and who does not as long as we meet the quota. Most people would not consider this to be fair either. Maybe the right *amount* of prisoners would get released, but at least some more dangerous criminals would be granted parole at the expense of some less dangerous criminals who are less likely to re-offend.

[1] Note that the COMPAS algorithm is typically framed the other way around, with parole being granted as the negative outcome and parole being denied as the positive outcome. There is nothing wrong with this - positive and negative are just statistical terms. However, binary classifiers are typically modelled with the 'good' outcome as the positive outcome. The terms used for COMPAS have been adjusted to reflect this and to avoid confusion later on.

On the other hand, it is possible to imagine situations where this kind of fairness quota would be helpful. If we were looking to recruit parents on to a school board, we might select an equal ratio of men and women. Likewise, if we were recruiting for a graduate training scheme, we might want to make sure that three or four out of every twenty places go to people of colour to reflect the United Kingdom's racial demographics. As well as achieving the desired representation, these decisions would enable a more extensive selection of voices and ideas to be heard, benefiting both the school board and the training scheme. The type of unfairness we are trying to prevent here is called *representative harm*, and the relevant fairness metric is *demographic parity*.

Returning to recidivism, it seems like the decision to grant a prisoner parole should be influenced by their risk score. Unlike with the school board, here we are primarily looking to avoid *allocative harm*: 'when a system allocates or withholds a certain opportunity or resource'. In contrast, representative harms 'occur when systems reinforce the subordination of some groups along the lines of identity' and 'can take place regardless of whether resources are being withheld'[12].

As we cannot look into the future to see if people will re-offend or not, we must predict this as best as possible by using aggregate data from past prisoners. Mapping all the social, psychological and developmental characteristics that cause an individual to re-offend is challenging. Besides simply using the wrong features, we might end up using features biased against specific demographics. Arguably, this is what the COMPAS algorithm did by not considering that a prior history of drug offences was more common for black people than white people, despite these demographics taking drugs at similar rates[32]. On the other hand, leaving those features out altogether might decrease the algorithm's overall accuracy, potentially harming all prisoners.

Despite these weaknesses, using a measure of likeliness-to-re-offend, however flawed, is much better in this case than granting parole by quota. Further, suppose we observe that the proportion of true positives (prisoners granted parole who do not re-offend) is lower for black prisoners than white prisoners. In that case, we could adjust the algorithm so that the true positive rates are equal. This type of fairness is called *equality of opportunity*.

Later on, we will look at the downsides to demographic parity and equality of opportunity and why we might derive benefits from using a weighed balance of both metrics. However, to interpret these metrics correctly, we must first examine how a binary classifier's results can be formally evaluated and how we can use these statistics to create different fairness metrics.

2.3 Evaluating a binary classifier

	$\hat{Y} = 1$	$\hat{Y} = 0$
$Y = 1$	TP true positive	FN false negative
$Y = 0$	FP false positive	TN true negative

Table 2.1: Binary classifier confusion matrix

The above table is a confusion matrix (adapted from Berk *et al.*)[7]. It shows all the possible results from a binary classifier. Y represents the binary ground truth value $\{0, 1\}$ - that is, the label that came with the data before classification. $\hat{Y} \in \{0, 1\}$ is the classification label given to the data by the algorithm. There are four possible outcomes:

A *true positive* is when the classifier correctly predicts a value of 1:

TP = $\hat{Y} = 1$ and $Y = 1$

A *true negative* is when the classifier correctly predicts a value of 0:

TN = $\hat{Y} = 0$ and $Y = 0$

A *false positive* is when the classifier incorrectly predicts a value of 1:

FP = $\hat{Y} = 1$ and $Y = 0$

A *false negative* is when the classifier incorrectly predicts a value of 0:

FN = $\hat{Y} = 0$ and $Y = 1$

Below are some of the standard metrics used to evaluate bias in a binary classification algorithm: others that are less relevant for the current work are omitted. Each metric is illustrated with two example cases: recidivism risk and loan applications. Unless stated otherwise, all definitions are taken from Verma and Rubin[34].

The COMPAS algorithm is not a binary classifier, as it outputs recidivism risk scores for inmates from 0 to 10[7]. (It is technically an n-ary classifier where n = 10.) For the sake of this exposition, we will discuss a hypothetical binary classifier that classifies prisoners as low risk (1) or high risk (0). However, it is worth noting that all of these metrics can easily be extended to n-ary classification problems.

Overall procedure accuracy

$\frac{TP+TN}{TP+FN+FP+TN}$ or $p(\hat{Y} = 1 \text{ and } Y = 1) + p(\hat{Y} = 0 \text{ and } Y = 0)$

Overall procedure accuracy represents the percentage of correct decisions made by the classifier.

- For COMPAS, this would be the number of prisoners correctly predicted not to re-offend plus the number of prisoners correctly predicted to re-offend divided by the total number of prisoners.
- For loan applications, this would be the number of people who meet the criteria for a loan and are granted a loan plus the number of people who do not meet the criteria for a loan and are denied a loan, divided by the total number of applicants.
- When the literature (and this paper) refers to accuracy in a binary classifier, overall procedure accuracy is what is meant.

Predictive values

Positive predictive value: $\frac{TP}{TP+FP}$ or $p(Y = 1|\hat{Y} = 1)$

Negative predictive value: $\frac{TN}{TN+FN}$ or $p(Y = 0|\hat{Y} = 0)$

- For COMPAS, the percentage of inmates labelled as low-risk who do not re-offend and the percentage of inmates labelled as high-risk who do re-offend, respectively.
- For loan applications, the percentage of people granted a loan who met the criteria and the percentage of people denied a loan who did not meet the criteria, respectively.

Base rates

Positive base rate: $\frac{TP+FN}{TP+FN+FP+TN}$ or $p(Y = 1)$

Negative base rate: $\frac{FP+TN}{TP+FN+FP+TN}$ or $p(Y = 0)$

Base rates measure the ground truth of the positive and negative conditions in a classifier. The positive and negative base rates sum to 1.

- For COMPAS, the percentage of ex-convicts who do not re-offend and the percentage who do re-offend.
- For loan applications, this would be the percentage of people who meet the criteria for a loan and the percentage who do not.
- Like with COMPAS, decision-makers cannot tell with certainty who will pay their loans back in the future, so they have to predict this using factors like age, employment status and credit history.

Prediction distributions

Positive prediction distribution: $\frac{TP+FP}{TP+FN+FP+TN}$ or $p(\hat{Y} = 1)$

Negative prediction distribution: $\frac{TN+FN}{TP+FN+FP+TN}$ or $p(\hat{Y} = 0)$

In contrast to base rates, prediction distributions measure the percentage of positive and negative decisions outputted by the classifier. Positive and negative prediction distributions sum to 1.

- For COMPAS, this would be the percentage of inmates predicted not to re-offend and the proportion predicted to re-offend, regardless of the ground truth values.
- For loan applications, this would be the percentage of people granted a loan and the percentage of people denied a loan, irrespective of the ground truth values.

As the prediction distribution does not rely on the ground truth values, an algorithm could output a high positive prediction distribution simply by classifying everyone in the ground truth negative condition as positive.

True positive rate

$\frac{TP}{TP+FN}$ or $p(\hat{Y} = 1|Y = 1)$

The number of correct positive predictions made by the classifier, divided by the total number of ground truth positives.

- For COMPAS, the number of low-risk prisoners correctly granted parole, divided by the total number of low-risk prisoners.
- For loan applications, the number of people granted a loan who met the criteria, divided by the total number of people who met the criteria.
- The true positive rate is also referred to as an algorithm's *sensitivity*.

False positive rate

$\frac{FP}{FP+TN}$ or $p(\hat{Y} = 1|Y = 0)$

The number of incorrect positive predictions made by the classifier, divided by the total number of ground truth negatives.

- For COMPAS, the number of high-risk prisoners mistakenly granted parole, divided by the total number of high-risk prisoners.
- For loan applications, the number of people granted a loan who did not meet the criteria, divided by the total number of people who did not meet the criteria.

True negative rate

$\frac{TN}{TN+FP}$ or $p(\hat{Y} = 0|Y = 0)$

The number of correct negative predictions made by the classifier, divided by the total number of ground truth negatives.

- For COMPAS, the number of high-risk prisoners correctly denied parole, divided by the total number of high-risk prisoners.
- For loan applications, the number of people denied a loan who did not meet the criteria, divided by the total number of people who did not meet the criteria.
- The true positive rate is also referred to as an algorithm's *specificity*.

False negative rate

$\frac{FN}{FN+TP}$ or $p(\hat{Y} = 0|Y = 1)$

The number of incorrect negative predictions made by the classifier, divided by the total number of ground truth positives.

- For COMPAS, the number of low-risk prisoners mistakenly denied parole, divided by the total number of low-risk prisoners.
- For loan applications, the number of people denied a loan who did meet the criteria, divided by the total number of people who met the criteria.

2.4 Fairness metrics

This study will focus on group fairness, where we evaluate and compare fairness metrics based on subgroups rather than individuals. Although individual fairness is advocated for by some researchers, we will see that this is not a solution for our situation. Adel *et al.* summarise group fairness as 'the outcome of an automated decision-making system should not discriminate between subgroups characterised by sensitive attributes such as gender and race'[2].

Consider a data set that can be partitioned into several subgroups. Each subgroup is a subset of the population based on a protected attribute such as race or sex. Let us use sex as the protected subgroup for the loan example and race for the COMPAS example (since this work is restricted to datasets that can be partitioned into two subgroups, we will restrict the COMPAS example to black and white prisoners).

2.4.1 Demographic parity

Demographic parity is achieved when the *positive prediction distribution* is the same for each subgroup.

$$p(\hat{Y} = 1|s \in S_1) = p(\hat{Y} = 1|s \in S_0) \tag{2.1}$$

where $S_1, S_0 \subset S$ are subgroups that partition the dataset[7].

When evaluating a binary classification for fairness, it is conventional to let S_1 stand for the privileged subgroup. The privileged subgroup is the subgroup that is more likely to have some social, cultural or historical advantage in this particular decision-making problem.

The demographic parity score (DP score) is defined as:

$$DP_{score} = p(\hat{Y} = 1|s \in S_1) - p(\hat{Y} = 1|s \in S_0) \tag{2.2}$$

If the two subgroups have the same positive prediction distribution, the DP score will be 0, and demographic parity is achieved. A positive score means that the positive prediction distribution is higher for the privileged subgroup S_1. A negative score means that the positive prediction distribution is higher for the less-privileged subgroup S_0. Since the positive prediction distribution can vary from 0 (no positive classifications in a subgroup) to 1 (every classification in a subgroup is positive), the DP score can vary from -1 to 1.

- For COMPAS, a DP score of 0 would mean that an equal proportion of white and black prisoners are granted parole. If there are twice as many white prisoners as black prisoners, twice as many white prisoners should be granted parole. Whether the prisoners are high- or low-risk is not essential in this decision, although the information could be used as long as the quota is met.
- For loan applications, a DP score of 0 would mean that an equal proportion of men and women are given loans, regardless of if one sex has a higher base rate (i.e. a higher percentage of one sex meet the criteria for a loan).

There are significant issues with demographic parity. It has been shown that women have a lower recidivism rate than men, all other factors being equal[11]. A recidivism prediction model tuned to achieve demographic parity between men and women would systemically overestimate women's recidivism rate, resulting in unfair and 'unnecessarily harsh judicial decisions'[11][7]. Hardt *et al.* concur that if a protected attribute is correlated with the ground truth outcome Y, there is a trade-off between demographic parity and accuracy[18].

On the other hand, we have seen how demographic parity is helpful in contexts where the priority is to ensure a representative sample of people with different protected characteristics. However, there are problems even in these situations. Let us assume for the sake of exposition that women are more qualified to serve on a school board than men. Even if qualifications only matter slightly in this scenario, the people in charge of hiring for our hypothetical school board might be accused of tokenism. Women may be angry that they were not given a fair chance (they would implicitly be using equality of opportunity as their fairness metric here). The men might feel uncomfortable being a target of criticism and may suffer from poor performance if they are underqualified. If representation and allocative fairness are both important, it might be a good idea to try a weighted combination of demographic parity and equality of opportunity. The weights might shift depending on the balance of opinions of the various parties involved.

2.4.2 Equality of opportunity

Equality of opportunity is achieved when the *true positive rate* is the same for each subgroup.

$$p(\hat{Y} = 1|Y = 1, s \in S_1) = p(\hat{Y} = 1|Y = 1, s \in S_0) \tag{2.3}$$

where $S_1, S_0 \subset S$ are subgroups that partition the dataset[18].

The equality of opportunity score (EO score) is defined as:

$$EO_{score} = p(\hat{Y} = 1|Y = 1, s \in S_1) - p(\hat{Y} = 1|Y = 1, s \in S_0) \tag{2.4}$$

If the two subgroups have the same true positive rate, the EO score is 0, and equality of opportunity is achieved. A positive score means that the true positive rate is higher for the privileged subgroup S_1. A negative score means that the true positive rate is higher for the less-privileged subgroup S_0. Since the positive prediction distribution can vary from 0 (every ground truth positive classified as a negative) to 1 (every ground truth positive classified as a positive), the EO score can vary from -1 to 1.

In the loan application case, an EO score of 0 means an equal chance that qualified men and qualified women receive a loan. Crucially, this differs from demographic parity because it allows differences in the base rates to be accommodated. For instance, imagine that 100 men and 100 women apply for loans. Sixty women are qualified for the loan, but only 45 men are. If an algorithm decides that 48 of these women and 36 men receive a loan, then the true positive rate for both sexes is 80 per cent, and equality of opportunity has been achieved.

Note that for equality of opportunity, it does not matter if the false positive rates are different. Had, say, 12 of the 40 unqualified women and 24 of the 55 unqualified men also received a loan, this would mean that 60 men and 60 women received a loan, and this toy example would also satisfy demographic parity. In practice, it is implausible that demographic parity and equality of opportunity will hold simultaneously. Moreover, you could not describe this situation as 'total' fairness, as it could be perceived as unfair to women that a higher proportion of men (43.6%) than women (30%) received a loan despite being unqualified.

There is a type of allocative decision-making situation where you might want to use a weighted balance of demographic parity and equality of opportunity. Redistributive justice involves making up for systematic injustice historically endured by protected subgroups that may have unfairly affected their chances of being allocated to the positive outcome[13]. However, this must be counterbalanced with the desire to choose people who deserve on paper to, say, get the job or be let out of prison. A balance of demographic parity and equality of opportunity could result in more false positives for the protected subgroup, which might be the right decision if you believe the subgroup deserves more of a chance than their ground truth ratings suggest.

As well as being a technical choice, redistributive justice is a contentious political decision. Stakeholders may have different opinions about how 'far' redistributive justice should go and even if it should be used at all. Reaching a compromise between these alternative opinions would determine the weighting of the two fairness metrics.

The use of any fairness metric needs to be evaluated in context to understand why it is being used, what it is really measuring and who stands to benefit by its use.

2.4.3 Additional fairness metrics

The following fairness measures are not looked at in this study but are presented to demonstrate some of the various alternative metrics that researchers have developed. Many more are not listed. For example, Verma and Rubin detail more than twenty different metrics used to evaluate a binary classifier. The researchers looked at a credit score dataset and observed gender bias when using some of these measures. However, this bias was not present for every measure, further demonstrating the importance of metric selection to suit a particular problem[34].

False positive error rate balance is similar to equality of opportunity, except it is achieved when the *false positive rate* is the same for each subgroup. It means that the classifier's probability of assigning the positive outcome to people in the ground truth negative category should be the same across subgroups[34].

Equality of odds is achieved when the *true positive rate* and the *false positive rate* is the same for each subgroup. It can be thought of as a stricter extension of equality of opportunity[18].

Predictive parity is achieved when the positive predictive value is the same for each subgroup. Out of all the classifier's positive assignations, the proportion of ground truth positives should be the same across subgroups[34].

2.5 The impossibility of total fairness

For an algorithm to achieve total fairness, all of these criteria must be met (along with additional metrics beyond this paper's scope). However, Chouldechova has shown that total fairness cannot be met when the base rate of the ground truth value Y (e.g. rate of recidivism, rate of loan repayment) differs between subgroups[10]. More specifically, she showed that if the subgroups' base rates differ, it is impossible to have equal false positive rates, false negative rates *and* equal accuracy rates for the ground truth positive values (positive predictive values)[7]. Similar impossibility results can be derived for other fairness metrics[24]. Unless we are dealing with trivial cases where the base rates of the subgroups are identical, we are therefore compelled to make trade-offs when we implement fairness metrics.

Chouldechova states the following equation:

$$FPR = \frac{PBR}{1-PBR}\frac{1-PPV}{PPV}(1-FNR) \tag{2.5}$$

where FPR is the false positive rate, PBR is the positive base rate, PPV is the positive predictive value and FNR is the false negative rate. This equation clearly shows how a change in the positive base rate for one subgroup would affect the other three values. The proof is straightforward (reframing Mowbray in probabilistic terms)[23]:

$$\begin{aligned}
\frac{PBR}{1-PBR}\frac{1-PPV}{PPV}(1-FNR) &= \frac{p(Y=1)}{p(Y=0)}\frac{p(Y=0|\hat{Y}=1)}{p(Y=1|\hat{Y}=1)}p(\hat{Y}=1|Y=1) \\
= \frac{p(Y=1)}{p(Y=0)}\frac{p(\hat{Y}=1|Y=0)p(Y=0)}{p(\hat{Y}=1)}\frac{1}{p(Y=1|\hat{Y}=1)}\frac{p(Y=1|\hat{Y}=1)p(\hat{Y}=1)}{p(Y=1)} \\
&= p(\hat{Y}=1|Y=0) \\
&= FPR
\end{aligned} \tag{2.6}$$

The second line is derived by use of Bayes' theorem.

Berk *et al.* reach a similar conclusion, stating that 'except in trivial cases, it is impossible to maximise accuracy and fairness at the same time, and impossible simultaneously to satisfy all kinds of fairness'. We must, therefore, 'consider challenging tradeoffs'[7]. Similarly, Northpoint cites the differing base rates of recidivism amongst prisoners of different races as an explanation for the COMPAS algorithm's divergence in false positive and false negative rates[15]. However, we must remember that ground truth rates can be deceptive. Data collection biases might disproportionately affect disprivileged subgroups (for example, white people may live in neighbourhoods that are patrolled less by police, or they may commit types of crimes that they are less likely to be caught for). There is also a more fundamental issue: are the base rates different because they say something intrinsic about a population or because poverty and discrimination have created adverse outcomes?

Most work has only looked at the trade-off between one definition of fairness and accuracy. We have shown that there are situations that could benefit from a weighted combination of two fairness metrics. Even though this does not guarantee total fairness, it might get us closer to balancing the different types of fairness needed for complex decision-making processes in the real world.

The current work will test the hypothesis that when a trade-off between two fairness metrics is merited, training a model with a composite loss function that optimises both metrics simultaneously is a viable alternative to training two separate models for each metric and combining them as an ensemble model.

2.6 Solutions to the fairness problem

Many solutions to the problem of algorithmic bias have been proposed. There are three techniques: pre-processing, in-processing and post-processing. In pre-processing, the data is modified before the binary classifier is trained. In-processing involves changes made directly to the classifier in order to make its results fairer. Finally, post-processing modifies the results of the binary classifier without changing the classifier or the data beforehand.

We will look first at how fairness cannot be obtained by the seemingly simple solution of removing protected attributes. Then we will look at the concept of individual fairness and show why it is ineffective in combating unfairness in algorithmic decision-making. We will then evaluate adversarial fairness approaches that use different model architectures to remove bias during the in-processing or pre-processing stages.

2.6.1 Removing protected attributes

One proposed solution to algorithmic bias is to remove protected attributes from the feature vector. This would be a type of pre-processing approach. However, as we have seen with COMPAS, other features often act as proxies for protected attributes. We could remove all the proxies as well, but this would be questionable for two reasons. It has been argued that many if not all features are 'at least partially correlated with protected group status'[11]. Additionally, removing many features is likely to lower the algorithm's accuracy substantially[3]. Given the limitations of this method, this study will not involve any modification of the data to remove protected attributes or their proxies.

2.6.2 Individual fairness

Corbett-Davies and Goel recommend abandoning fairness metrics altogether due to the trade-offs incurred when using them. Instead, they argue that 'constructing risk scores that best capture individual risk', including the use of any protected attributes correlated to the outcome, calculating individual risk scores and implementing a 'threshold' score that determines the outcome, results in an optimal outcome where 'similarly risky individuals are treated similarly, regardless of group membership'[11]. Effectively, this is

an endorsement of the method used by COMPAS (indeed, it goes further by allowing the use of protected attributes). It is an example of a post-processing approach, as the threshold is set after the model is trained. However, this proposal eludes several important issues.

First, it does not account for potential bias against one (or more) subgroups embedded into the training data, which, as previously seen, may "encode prior prejudicial or biased assessment"[6]. Crucially, even with pristine data, a minority subgroup might meet the threshold less often because 'when minority groups do not follow the same pattern of behaviour as the majority group, machine learning may struggle to model the behaviour of the minority as effectively as the majority because there will be proportionally fewer examples of the minority behaviour from which to learn'[6]. Threshold policies based on individual fairness do not address the problems at the heart of algorithmic bias and will not be considered in this study.

The following two sections will consider ways to change a machine learning model's architecture to encourage it to produce fairer decisions (in the sense of group fairness). These can be in-processing approaches, as in Adel *et al.*'s direct modification of a binary classifier[2]. They can also be pre-processing approaches, where generative models are modified for fairness and trained on the original data, producing generated data used for fairer downstream classification[30][35].

2.6.3 One-network adversarial learning

Adel *et al.* adapt an existing classifier by adding an adversary as the second-to-top layer of the model. Above this layer, they use two output layers for classification. One produces the classification prediction, while the second tries to predict the protected attribute from the other feature vectors. The adversary layer 'aims at maximising the performance of the label predictor, while minimising the performance of the sensitive [protected] attribute predictor by reversing the gradients of the latter' during backpropagation[2].

As is proposed in the current work, the loss function simultaneously optimises for classification accuracy and two fairness metrics: in this case, demographic parity and disparate mistreatment. They define *disparate mistreatment* as the difference between the false positive rates of subgroups plus the difference between the false negative rates of subgroups; the inverse of Hardt *et al.*'s equality of odds[2][18].

The model was trained on the COMPAS data set and achieved a significant improvement in both metrics while maintaining an overall procedure accuracy rate of less than one percentage point lower than a model trained on COMPAS that only optimised accuracy. Results of almost such a high quality were found on a dataset predicting income from fourteen features using sex as the protected attribute, showing their method is generalisable within the class of low-dimensional tabular feature vectors.

While it is very encouraging that this paper demonstrates that two fairness metrics can be evaluated using one model, it does not attempt to analyse whether the model was fairer than the average of two similar models that had been optimised solely for one fairness metric. It also does not investigate how the trade-off between different types of fairness can be parameterised by weighting the loss function, leaving scope for the present work.

Additionally, the two data sets used low-dimensional feature vectors, whereas this study will investigate high-dimensional multimedia data. It cannot be assumed that these results would map to high-dimensional data, given the vastly increased number of features in the feature vector and the different nature of image classification.

Finally, the major weakness of using an in-processing approach is that we need to access the classifier when training to implement the fairness solution. This approach is not feasible in all machine learning implementations, where the model might need to be treated like a 'black box' by somebody who is not a specialist. This situation is increasingly the case as machine learning models permeate more and more areas of society. A pre-processing approach is model-agnostic and also permits access to sensitive attributes at the time of training.

We seek another pre-processing approach that does not remove the protected attributes or their proxies and produces a modified data set that is fairer with respect to the chosen fairness metrics. We have chosen to produce a generated dataset using a generative model.

2.6.4 Fairness GAN

A generative adversarial network (GAN) is a type of adversarial learning network where a noise vector ('fake' data) is fed into the generator component. A discriminator then receives the real data and the fake data and attempts to predict which is which, while the generator tries to fool the discriminator into saying the fake images are real. Over time, the network reaches an equilibrium where the discriminator cannot discern between the real and fake data, and the generator output comes to approximate the statistical distribution of the feature vector of the real data set[17].

A modification of the GAN, the auxiliary classifier GAN (ACGAN), includes a second classifier in the top layer of the discriminator network[25]. As in Adel *et al.*, this classifier is designed so that it is penalised if it correctly predicts the protected attribute.

The best example of current research in GAN fairness is 'Fairness GANs' by Sattigeri *et al.* They train one modified AC-GAN to optimise demographic parity and a separate one to optimise equality of opportunity. They test their models with three different image-based data sets. They found that the model built to optimise demographic parity improved demographic parity with respect to a GAN trained solely to optimise accuracy. Equally promising results were obtained for the model built to optimise equality of opportunity.[30]. These results demonstrate that GANs can generate a dataset that makes a binary classifier trained on it fairer regarding chosen metrics.

The current paper draws heavily on Sattigeri et al. The architecture and loss functions they use and how they were partially replicated in the current work are discussed in the next chapter. Although we did not end up using the same datasets as Sattigeri et al., the results they obtained can also be compared with the results of this work.

The fact that these papers both use adversarial training demonstrates a general method for improving fairness for specified metrics. Naturally, the question arises whether the combination of fairness metrics demonstrated in Adel *et al.* is possible to implement in GANs. This work shows that it is indeed possible to parameterise two loss functions to achieve a weighed balance between two fairness metrics using GANs.

2.7 Fairness and faces

The decision was made to look at fairness in decision-making based on images - and in particular, images of faces - for several reasons. First, it posed an interesting intellectual challenge with the opportunity to learn a lot about deep neural networks. Second, as already observed, the literature is heavily focused on algorithmic bias in situations with low-dimensional tabular data. On discovering Adel *et al.*'s paper combining fairness metrics in low-dimensional data and Sattigeri *et al.*'s paper showing the plausibility of using fairness metrics to evaluate image classification, it was clear that there was a possible research gap in trying to combine fairness in image data.

Faces are intrinsically interesting to us as human beings trying to make sense of the social environment around us. They also have particularly fascinating use cases in the domain of algorithmic fairness. The most prescient example is facial recognition algorithms. The decision made by a facial recognition algorithm is one or more of the following:

- Does this image contain a person's face or not?
- Does this image contain a face belonging to a person of a particular subgroup?
- Is the person we are looking for in this photo?

It is clear how an individual could face severe and unjust consequences if the false positive rate were higher for a demographic they belonged to. It is well documented that facial recognition algorithms have a lower accuracy rate for black people and other racial minorities. Some people have embraced this, arguing that they do not wish to be detected by these algorithms. However, this ignores the fact that lower accuracy could mean a higher false positive rate. This can result in the wrongful arrest of innocent people because the algorithm incorrectly identified them as the perpetrator of a crime[19]. However, it could also cause difficulties in more trivial (but still important) situations like unlocking your phone with facial recognition software in bad lighting.

Like most tools, facial recognition can also be a force for good[2]. In this case, we would be more concerned about false negatives: cases where the algorithm should recognise a face but fails to do so (or believes it is somebody different). CENSER is a program using facial recognition technology in India to identity and rescue trafficked girls[31]. It would be a dire situation if a girl went undetected because some aspect of her appearance was misclassified more often by a facial recognition algorithm. From where do these biases originate? Even if the data collection is unbiased, there will still be fewer samples collected from minority subgroups. Consequently, there will be fewer examples for the model to train on, and it may learn less well than on a majority subgroup. There may also be selection biases, such as the minority subgroup being warier about researchers using images of their faces in new datasets. Sampling biases may mean that data from the majority subgroup is over-represented[14]. There may even be biases embedded in the equipment used to record data, such as the infamous Shirley Cards used to calibrate skin tone in photography that did not photograph people of colour as accurately as white people[29].

While we should endeavour to reduce data bias as much as possible, it is nearly inevitable that some will remain in any dataset. Accordingly, the current work investigates ways to generate new datasets that can reduce these biases in binary classification.

[2]Properly speaking, this should read 'facial recognition is also deployed in situations where most people would agree it was a beneficial use of the technology'. It is important to remember that facial recognition in policing can also help detect real criminals - there are two sides to every story.

Chapter 3

Methodology

3.1 Research questions

- Is it possible to build a combined fairness GAN that can achieve a balance between demographic parity and equality of opportunity? Can this balance be parameterised in a binary classifier to achieve the desired trade-off between the two fairness metrics?
- Is there a net fairness cost in implementing two fairness metrics? Specifically, let total bias be the sum of the demographic parity score and the equality of opportunity score. How does total bias in a combined fairness model compare to the average of total bias in a model optimised for demographic parity and one optimised for equality of opportunity model?
- Does implementing combined fairness cause a significant trade-off with the overall procedure accuracy of the classifier? To what extent might this weaken the findings?
- How does the combined fairness GAN achieve its results? Are any different types of unfairness incurred in optimising for demographic parity and equality of opportunity?
- What types of decision settings would be aided by a model that uses a combined fairness approach? What stakeholders might benefit from this type of model?

3.2 Framework

3.2.1 Languages and Libraries

It was an easy decision to use Python for this project. As the purpose of this work is to research and test a novel concept rather than develop a web- or application-based piece of software, it was sufficient that the language ran in the chosen environment. Compatibility with other devices was not a factor. Python has by far the most extensive collection of machine learning libraries, along with a massive community of practitioners offering advice, support and tutorials.

A more critical choice was which machine learning library to use: TensorFlow or PyTorch. Before starting this project, I had a small amount of TensorFlow experience and none with PyTorch. Furthermore, the Keras wrapper would have provided a fast and easy way to code models. However, PyTorch was chosen because of its Pythonic structure, making it easy to integrate the numerous helper functions with the machine learning code. Additionally, the low-level functionality was vital for fine-tuning the loss functions for the fairness GANs.

Various libraries were used to streamline the process. Tensors were converted into NumPy arrays before saving. Pandas was used to read in the original dataset from a CSV file. Matplotlib was used to plot images of the generated data and learning curves for the various generators, discriminators and classifiers. TQDM was used to display a progress bar when training the GAN or the classifier. Finally, Pillow was used for converting between images and tensors.

3.2.2 Software and Environment

Jupyter Notebook is an easy-to-use, interactive browser-based environment primarily used for data science and machine learning. Code is divided into separate executable cells, which gives the enormous advantage of debugging and testing very rapidly. If you want to view the images outputted by a GAN or a diagram displaying learning rates, it is helpful to have these displayed right below the relevant cell in Jupyter. It is also straightforward to run a larger section of code if needed.

There are two significant downsides to using Jupyter. First, you cannot do breakpoints and step-through debugging like in an IDE. This absence meant that sometimes it was challenging to find the source of an error, especially when I was getting to grips with many new libraries.

Second, you cannot execute parts of a Jupyter notebook inside another notebook (to the best of my knowledge). Consequently, the whole code was contained in one file, which eventually grew to be very long. This problem was mitigated by dividing the notebook into collapsible sections, so I only needed to have the section I was working on open at a time.

Jupyter can be run locally using a CPU, which I did for the project's initial stages. Once I started training my GANs, it became clear that I needed a GPU. GPUs have become inextricably linked to machine learning due to their ability to quickly perform the hundreds of thousands of matrix calculations that form the basis of deep learning. I used Google's Colaboratory (Colab) cloud-hosted environment in order to access their free GPUs. Overall, Colab was very useful, and GAN training was around six times as fast using a GPU.

As it is a free service, Colab cuts you off from the GPU after several hours, and you have to wait up to half a day to use it again. This was mostly not an issue, but towards the end of my project, I needed more time on the GPU. At this point, I used notebook hosting on Paperspace Gradient. This service allowed me to run several notebooks in parallel. Since the GANs each took around two to four hours to train (depending on the GPU specifications), this was invaluable in saving time.

3.3 Data collection

Initially, the intention was to use two of the three datasets used by Sattigeri *et al.* in the Fairness GAN paper. This way, the classification results using the generated datasets constructed in this study could be directly compared to their results. However, upon closer inspection, two of the three datasets required substantial manual labelling, cropping and other forms of data cleaning and pre-processing.

The third data set - the Google Draw power outlet drawings - looked more suitable, as it appeared to be publically available. Unfortunately, it was unclear how to extract the protected attributes and the classification outcomes from the pre-processed version of the dataset. Looking again at Sattigeri *et al.*, it was clear that they had experienced a related issue and had chosen to download the raw data and clean this themselves. The amount of work required was beyond this project's capacity, so a different dataset was sought.

As the particular goal was to investigate unfairness in faces, Liang *et al.*'s SCUT-FPB-5500 benchmark dataset of male and female faces was selected[20]. The dataset was publicly available and did not need pre-processing beyond scaling, converting from JPEG to PyTorch tensor and extracting the protected

attributes and outcomes from the file names.

SCUT-FPB-5500 can be analysed by sex or by race (the dataset contains Asian and white faces). However, as there were 4000 Asian faces and only 1500 white faces, the dataset was not balanced by class. Although this is often the case in the real world when evaluating subgroup fairness, the imbalance complicates the training process. As there were 2750 male and 2750 female faces, sex was chosen as the protected attribute.

A total of 60 crowdsourced volunteers labelled each of the 5500 faces' attractiveness using a scale from 1 to 5. The average of these sources was provided with each image[20]. We thresholded scores of 2.85 and above to 1 (attractive) and scores below 2.85 to 0 (not attractive) to create a balance of outcomes. Attractiveness is a suitable outcome variable for this research because it is an allocative decision with an aspect of judgement (attractive people may be favoured in some decision-making contexts), rather than a descriptive feature like 'has hair' or 'wears glasses'.

The purpose of Liang *et al.*'s paper is to create a face image benchmark, and there are thus are no classification results in the paper with which to compare ours. However, given that our hypothesis rests on a relative comparison between the different GANs we will build in this study, this is not necessarily a huge problem. On the other hand, a significant drawback of the present work is that there are no know results in other papers to compare ours directly. We will evaluate the results by comparing them to the Fairness GAN results, with the caveat that not using the same datasets significantly weakens this comparison.

Further, using a second dataset would have helped establish whether the conclusions drawn from SCUT-FPB-5500 were generalisable or whether the model had overfitted to the first dataset. These findings would have strengthened the hypothesis's external validity, allowing us to extrapolate the findings to different use cases more confidently. Therefore, one critical extension to this work would be testing the models on different datasets to see whether the conclusions still hold.

3.4 Formal statement of hypothesis

Let C be a family of binary classifiers trained using either a real dataset or a generated dataset produced by a Combined Fairness GAN trained on the real dataset.

A reminder that the overall procedure accuracy is defined as:

$$A = p(\hat{Y} = 1 \text{ and } Y = 1) + p(\hat{Y} = 0 \text{ and } Y = 0)$$

This can also be written as:

$$\frac{\text{number of correct classifications}}{\text{total number of items}} = \frac{TP + TN}{TP + FP + TN + FN}$$

A can vary between 0 (no correct classifications made) to 1 (every classification made is correct).

Also, recall the definitions of the demographic parity score and equality of opportunity score.

$$DP_{score} = p(\hat{Y} = 1 | s \in S_1) - p(\hat{Y} = 1 | s \in S_0)$$
$$EO_{score} = p(\hat{Y} = 1 | Y = 1, s \in S_1) - p(\hat{Y} = 1 | Y = 1, s \in S_0)$$

Let A_R be the overall procedure accuracy of the binary classifier C_R trained using the real dataset. Let R_{DP} and R_{EO} be the DP and EO scores of C_R when tested with the real test set.

Let FG_{DP} be a Combined Fairness GAN with a loss function component L_{DP} aimed at minimising the DP score. Let C_{DP} be a binary classifier trained on the data generated by FG_{DP}. Let DP_{DP} be the DP score of C_{DP} when tested with the real test set and let EO_{DP} be the EO score of C_{DP} when tested with the real test set. Finally, let A_{DP} be the overall procedure accuracy of C_{DP} when tested with the real test set.

Let FG_{EO} be a Combined Fairness GAN with a loss function component L_{EO} aimed at minimising the EO score. Let C_{EO} be a binary classifier trained on the data generated by FG_{EO}. Let DP_{EO} be the DP score of C_{EO} when tested with the real test set and let EO_{EO} be the EO score of C_{EO} when tested with the real test set. Finally, let A_{EO} be the overall procedure accuracy of C_{EO} when tested with the real test set.

Let FG_{CF} be a Combined Fairness GAN with a loss function component L_{CF} aimed at achieved a weighed balance between minimising the DP score and the EO score. Let C_{CF} be a binary classifier trained on the data generated by FG_{CF}. Let DP_{CF} be the DP score of C_{CF} when tested with the real test set and let EO_{CF} be the EO score of C_{CF} when tested with the real test set. Finally, let A_{CF} be the overall procedure accuracy of C_{CF} when tested with the real test set.

Let ϵ and δ be small real numbers and let α and β be real numbers.

There exists a Combined Fairness GAN with the following properties:

$$CF_{DP} + CF_{EO} < \frac{1}{2}(DP_{DP} + DP_{EO} + EO_{DP} + EO_{EO}) \tag{3.1}$$

$$R_{DP} > CF_{DP} > DP_{DP} \tag{3.2}$$

$$R_{EO} > CF_{EO} > EO_{EO} \tag{3.3}$$

$$A_R - \epsilon = A_{DP} \approx A_{EO} = A_{CF} + \delta \tag{3.4}$$

$$L_{CF} = \alpha(L_{DP}) + \beta(L_{EO}) \tag{3.5}$$

The null hypothesis is that a Combined Fairness GAN satisfying these properties does not exist.

Note that finding such a GAN in the time allocated is not proof that such a GAN does not exist. However, it would mean that we could not reject the null hypothesis at this time.

3.5 Models, architecture and loss

This section will explain how the different types of GAN relevant to this work function. It will also show how and why the GANs built for this project differ from these architectures.

3.5.1 Introduction to GANs

A GAN is a system of two artificial neural networks: a generator and a discriminator. The two networks compete against each other in an adversarial game that trains the generator to produce samples drawn from an approximation of the distribution of the training data. GANs can be trained on different types of data, but we will focus on images.

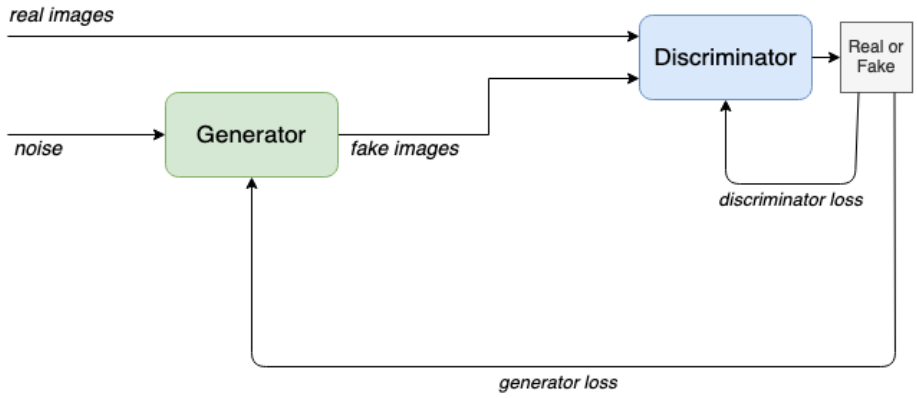


Figure 3.1: Basic GAN architecture

During training, the generator is seeded with random noise Z. This is fed forward through the neural network's various layers, resulting in a fake image $G(Z)$. The discriminator is alternately fed these generated images and real images X from the training set. The discriminator's goal is to distinguish between fake and real images. Crucially, we know whether the image is fake or real, and so we can respond to incorrect guesses by tuning the discriminator's weights through backpropagating the loss. Simultaneously, the generator is aiming to fool the discriminator into mislabelling its fake images as real. It uses the discriminator's feedback to adjust its weights through backpropagation to produce images that are more likely to fool the discriminator. Ideally, this process results in the two models reaching an equilibrium where the discriminator correctly guesses the image's source half of the time, and the generator can produce a range of good-quality images that look similar to (but are not copies of) the real images[17]. The training process will be examined in more detail in section 3.7.

Many loss functions can be used in a GAN, but for the sake of simplicity, we will be using the original loss function proposed by Goodfellow *et al.* in their seminal 2014 paper introducing GANs.[17]

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{x\sim p_z(z)}[\log 1 - D(G(Z))] \tag{3.6}$$

where G is the generator, D is the discriminator, $G(Z)$ is a generated fake image, $D(\cdot)$ is the discriminator's prediction about the image's source, p_{data} is the distribution of the real images and p_z is the noise distribution.

Observing that the first term cannot be influenced by the generator, this can be broken down as:

$$\max_D \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{x\sim p_z(z)}[\log 1 - D(G(Z))] \tag{3.7}$$

$$\min_G \mathbb{E}_{x\sim p_z(z)}[\log 1 - D(G(Z))] \tag{3.8}$$

The outputs of $D(\cdot)$ are in the range $[0, 1]$, with 0 meaning the discriminator is certain it is fake and 1 meaning it is certain it is real. The maximum value of a logarithm (regardless of base) between 0 and 1 is $log(1) = 0$. The minimum value is $log(0)$, which is asymptotic to $-\infty$. We therefore want $D(x)$ in equation 3.7 to be close to 1 and $D(G(Z)$ to be close to 0 (so that 1 - $D(G(Z) = 1$). This is achieved when the real image X is correctly classified as real and the fake image $G(Z)$ is correctly classified as fake. In equation 3.8, we want the discriminator to misclassify the fake image $D(G(Z)$ as real, so that 1 - $D(G(Z) = 1$ - $1 = 0$.

However, minimising the generator as in 3.8 leads to vanishing gradients and problems with training[**?**]. The following heuristic is used to circumvent this:

$$\max_G \mathbb{E}_{x \sim p_z(z)}[\log D(G(Z))] \tag{3.9}$$

Again, we want the discriminator to misclassify the fake image as real to maximise this term. In practice, using this heuristic means swapping the labels when training the generator[16].

Additionally, the PyTorch frameworks we work with are optimised for gradient descent (a minimisation problem) rather than gradient ascent (maximisation). Therefore, in practice, we minimise the binary cross-entropy loss of the distance between the source (real or fake) and the discriminator's prediction of the source. Binary cross-entropy is very similar to (the opposite version of) GAN loss and therefore models the problem well.

GANs are known to be challenging to train and suffer from a range of problems. If the generator gains the upper hand early on in the adversarial game, it will fool the discriminator using poor-quality images and will never be prompted to improve. Conversely, the discriminator may stymie the generator's growth by being too powerful. A common issue is mode collapse, where many randomly initiated noise vectors produce the same generated image. Unlike non-adversarial neural networks, there is no state of convergence for GANs to reach: 'with a GAN, every step taken down the hill changes the entire landscape a little. It's a dynamic system where the optimization process is seeking not a minimum, but an equilibrium between two forces'[9].

3.5.2 DCGAN architecture

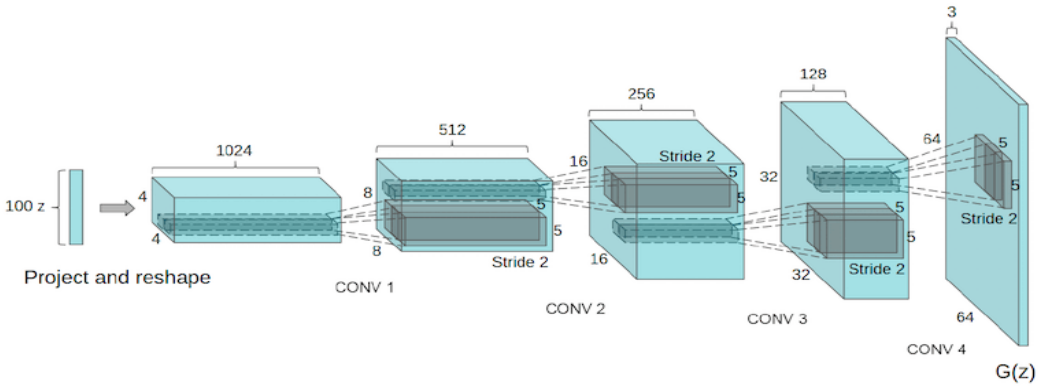


Figure 3.2: Transpose convolutions in the generator of a DCGAN

The Deep Convolutional GAN (DCGAN) is a popular and effective architecture for building GANs for image-related tasks. The generator and discriminator in the Combined Fairness GANs are based on the DCGAN architecture: designing a novel architecture would have been ineffective and unnecessary. The main difference between a basic GAN and a DCGAN is that the former uses fully connected linear layers to construct its models. In contrast, DCGANs use convolutional layers in the discriminator and transpose convolutional layers in the generator[27].

Convolutions work by passing a filter in strides across each channel of an image. The number of filters used determines how many channels the input of the next convolutional layer will have and how much the dimensions of the image are reduced. Rather than processing the image pixel-by-pixel, each filter learns to extract features from an image. The first layers extract lower-level features such as lines and

edges, while the higher layers combine these features to detect objects such as eyes and mouths.[1]

Transpose convolutions perform the opposite operations to reduce image channels while increasing image size. This can be seen in the above diagram taken from Radford *et al.*'s paper introducing the DCGAN[27].

3.5.3 Fairness GAN

The architecture of Sattigeri *et al.*'s Fairness GAN is much more sophisticated than the DCGAN. The generator used the noise vector and the conditional class label c to generate the image X using ResNet blocks and the outcome label y using linear layers. X and y were passed to the discriminator, which used the projection discriminator technique to project y onto X in order to calculate the joint source loss $p(S_J|X, y)$[30][22]. Also calculated were the standard source loss $p(S_X|X)$, the class-conditioned loss (for training stability[25]) $p(C = c|X)$ and the fairness loss term $p(C = c|y)$. The last term was maximised in both the generator and the discriminator, leading to a decorrelation of c and y that promoted demographic parity. For equality of opportunity, the fairness loss term was only used in the generator when the y output was equal to 1. Although we chose to decorrelate the outcome from the image, the concepts behind these loss functions greatly inspired the current work.

An attempt was made to replicate a simplified version of its structure using convolutional layers instead of ResNet blocks. This attempt was unsuccessful: the values of y produced by the generator stayed fixed regardless of the image generated. It was unclear whether this error was due to an incorrect implementation or a misinterpretation of what the authors had built (their code was not available to access).

In addition to the original GAN, the DCGAN and the Fairness GAN, the GAN and training loop design in this work were aided by Odena *et al.*'s paper on auxiliary classifier GANs[25], Hmrishav Bandyopadhyay's ACGAN tutorial[5], *Make Your First GAN with PyTorch* by Tariq Rashid[33] and Aladdin Persson's GAN tutorials[26].

3.6 Combined Fairness GAN

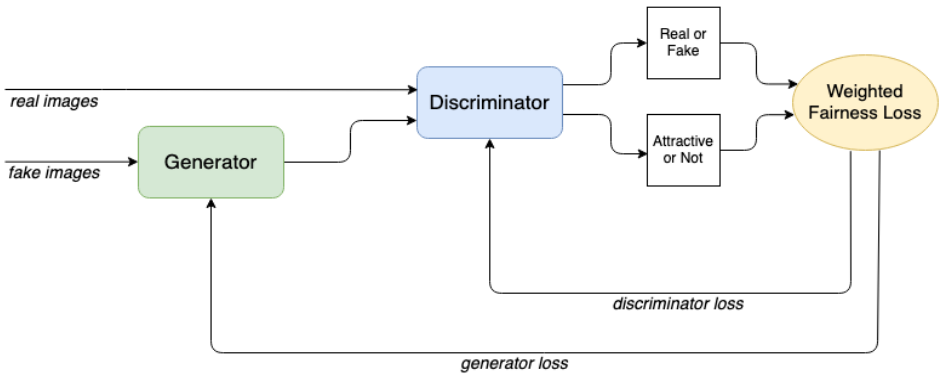


Figure 3.3: Combined Fairness GAN architecture

To distinguish it from the Fairness GAN, the models built and evaluated for this paper have been named Combined Fairness GANs. This section will look at the architecture of the generator and discriminator.

The diagram above shows the broad structure of the Combined Fairness GAN. It differs from the standard GAN in two ways. Firstly, it has an additional classifier at the top layer of the discriminator. This classifier

[1] In practice, feature extraction does not work as cleanly as this!

is structurally similar to the ACGAN and the Fairness GAN but is involved in predicting outcome, not class. Secondly, the loss function is weighted before the loss is calculated and backpropagated. The weighting depends on whether we are trying to minimise the demographic parity score, the equality of opportunity score or a weighted combination of both.

The loss functions for the generator and the discriminator is the average of the source loss (whether the image is real or fake) and the outcome loss (where the face is attractive or not). The rationale behind this choice of loss functions will be explained in the next section.

3.6.1 Discriminator

```
class Discriminator(nn.Module):
  def __init__(self, disc_features):
    super().__init__()

    # build model
    self.model = nn.Sequential(
      nn.Conv2d(3, disc_features, kernel_size=4, stride=2, padding=1),
      nn.LeakyReLU(0.2),
      self.block(disc_features, disc_features * 2, 4, 2, 1),
      self.block(disc_features * 2, disc_features * 4, 4, 2, 1),
      self.block(disc_features * 4, disc_features * 8, 4, 2, 1))

    self.model_source = nn.Sequential(
      nn.Linear(GAN_BATCH_SIZE * disc_features * 128, GAN_BATCH_SIZE),
      nn.Sigmoid())

    self.model_outcome = nn.Sequential(
      nn.Linear(GAN_BATCH_SIZE * disc_features * 128, GAN_BATCH_SIZE),
      nn.Sigmoid())

  def block(self, in_channels, out_channels, kernel_size, stride,
     padding):
    return nn.Sequential(
      nn.Conv2d(in_channels, out_channels, kernel_size, stride, padding,
          bias=False),
      nn.BatchNorm2d(out_channels),
      nn.LeakyReLU(0.2),
      nn.Dropout(0.5))

  # run model
  def forward(self, inputs):
    t = self.model(inputs).flatten()
    return self.model_source(t), self.model_outcome(t)
```

The discriminator takes in a PyTorch tensor representing a real or fake image with dimensions (GAN_BATCH_SIZE, 3, 64, 64). The 3 is the number of channels in an RGB image, and the images are 64 by 64 pixels. As in the DCGAN paper, disc_features is set to 64[27]. The image is passed through self.model, a sequence of convolutional layers that transforms the tensor to size (GAN_BATCH_SIZE, 512, 4, 4): a much smaller image with many more channels. The activation function during this stage of the model is LeakyReLU. Batch normalisation and dropout of 50% are also used for every layer in self.model except the first.

This intermediate representation is flattened to a one-dimensional tensor of size (GAN_BATCH_SIZE * 64 * 128), then passed through two separate linear layers to calculate the discriminator's predictions for the source of the image (real or fake) and the outcome (attractive or not). These linear layers both resize the

(GAN_BATCH_SIZE * 64 * 128) tensor to size (GAN_BATCH_SIZE) before using the sigmoid activation function, which rewards confident predictions and normalises the values to between 0 and 1. Each pair of (GAN_BATCH_SIZE) tensors returned represent the source and outcome values for the batch of images passed into the discriminator.

3.6.2 Generator

```
class Generator(nn.Module):
  def __init__(self, noise_size, gen_features):
    super().__init__()

    # build model
    self.model = nn.Sequential(
      self.block(noise_size, gen_features * 16, 4, 1, 0),
      self.block(gen_features * 16, gen_features * 8, 4, 2, 1),
      self.block(gen_features * 8, gen_features * 4, 4, 2, 1),
      self.block(gen_features * 4, gen_features * 2, 4, 2, 1),
      nn.ConvTranspose2d(gen_features * 2, 3, 4, 2, 1),
      nn.Tanh())

  def block(self, in_channels, out_channels, kernel_size, stride,
     padding):
    return nn.Sequential(
      nn.ConvTranspose2d(in_channels, out_channels, kernel_size, stride,
          padding, bias=False),
      nn.BatchNorm2d(out_channels),
      nn.ReLU())

  # run model
  def forward(self, inputs):
    return self.model(inputs)
```

The generator takes as input a noise vector of size (GAN_BATCH_SIZE, 128, 1, 1) and upsamples it in self.model using transpose convolutions. The noise vector contains normally distributed random values with mean 0 and variance 1. As recommended by the DCGAN paper, ReLU is used as the generator's activation function instead of LeakyReLU[27]. Batch normalisation and dropout of 50% are used for every layer in self.model except the last. The output from the final layer passes through a tanh activation function. This serves a similar purpose to the sigmoid activation but normalises the values from -1 to 1 to match the range of the real images' values. Like the batches of real images, the batches of generated images have size (GAN_BATCH_SIZE, 3, 64, 64).

3.7 GAN training loop

This section highlights some of the most important processes of the training pipeline. Code snippets are used to illustrate design decisions. Snippets have been edited for clarity and conciseness. The full code can be downloaded from the project GitHub: https://github.com/lizziehull/combined-fairness.

```
def train_gan(G, D, T, train_loader, epochs, text, save):

  loss_function = nn.BCELoss()

  optim_D = torch.optim.Adam(D.parameters(),
                             lr = 0.0001, betas=(0.5, 0.999))
  optim_G = torch.optim.Adam(G.parameters(),
                             lr = 0.0001, betas=(0.5, 0.999))
```

Binary cross-entropy loss is chosen as the loss function as for each of the two use cases (source = real or fake, output = attractive or not), there are two labels. The further away the discriminator's prediction is from the correct label, the more we want to penalise it. The generator trains indirectly through the discriminator's predictions. Adam is used for the optimiser as it is well-regarded. The learning rate of 0.0001 was arrived at by an informal parameter search.

For each epoch, we loop through every batch of the real train set. First, we decompose the batch into its component: a group of images X, a group of ground truth attractiveness labels y and a group of ground truth sex labels s.

```
      # train D on a batch of reals
      DR_source, DR_outcome = D.forward(X)
      DR_loss_source = loss_function(DR_source,
                                     torch.ones_like(DR_source))
      DR_loss_outcome = loss_function(DR_outcome, y)
```

We first train the discriminator on real images from the real train set. We pass a real image X through the discriminator and get back three predictions: whether the image is real or fake, whether the image is male or female and whether the image is attractive or not. Note that we know the ground truth for these predictions: the image is real (1), the class is s, and the outcome is y. We pass in these values and the predicted values to the loss function.

```
      # train D on a batch of fakes
      gen_image = G.forward(torch.randn(GAN_BATCH_SIZE, NOISE, 1, 1))
      targets_outcome = random_classes()
      DF_source, DF_class, DF_outcome = D.forward(gen_image.detach())
      DF_loss_source = loss_function(DF_source,
                                     torch.zeros_like(DF_source))
      DF_loss_class = loss_function(DF_class, targets_class)
      DF_loss_outcome = loss_function(DF_outcome, targets_outcome)
```

We generate an image by passing a tensor with randomly generated normally distributed values through the generator. The output is a generated image that we pass through the discriminator. We detach the gradients from the fake image before passing it through, as we only want to train the discriminator here and not the generator. This code is similar to that for the reals, except we pass in a tensor of zeros to the source loss because now the image is fake. We also do not have ground truth values for the fairness loss, so we pass in `targets_outcome`, a 1D GAN_BATCH_SIZE tensor of randomly generated 0s and 1s.

```
      loss_DR = (DR_loss_source + DR_loss_outcome) / 2
      loss_DF = (DF_loss_source + DF_loss_outcome) / 2
      loss_D = (loss_DR + loss_DF) / 2

      D.zero_grad()
      loss_D.backward()
      optim_D.step()
```

We now add the source and fairness losses together for the real and the fake batches, then add them together (dividing by the number of terms to normalise). The discriminator's gradients are cleared, the gradient is calculated, and the optimiser updates the model's weights and biases by taking an appropriately-sized step in the right direction.

```
      # train G using D loss
      # don't use detach here so gradient flows
      gen_image = G.forward(torch.randn(GAN_BATCH_SIZE, NOISE, 1, 1))
      targets_outcome = random_classes()
      G_source, G_outcome = D.forward(gen_image)
      G_loss_source = loss_function(G_source, torch.ones_like(G_source))

      if text == 'EO':
        G_outcome, targets_outcome = fairness_loss(G_outcome,
            targets_outcome)
      elif text == 'CF':
        G_outcome, targets_outcome = fairness_loss(G_outcome,
            targets_outcome)

      G_loss_outcome = loss_function(G_outcome, targets_outcome)

      loss_G = (G_loss_source + G_loss_outcome) / 2

      G.zero_grad()
      loss_G.backward()
      optim_G.step()
```

The generator is trained by creating a second fake image batch and forwarding it through the discriminator. Unlike when training the discriminator, the fake image batch's gradients are not detached before being passed through. Consequently, all updates made to the discriminator in this training step will backpropagate through the generator. Using a separate optimiser for the generator means that we can adjust the generator's weights while leaving the discriminator unchanged. Because we are using the heuristic of maximising $\mathbb{E}[\log D(G(Z))]$ instead of minimising $\mathbb{E}[\log 1 - D(G(Z))]$, we have to flip the labels we use when training the generator[16]. Therefore, we pass a tensor of ones (i.e. real) as ground truth source values to the source loss even though the source is fake.

The generator learns how to convince the discriminator that the fake images are real using the backpropagated gradient from the discriminator to learn how to make plausible images. This outcome should not lead to the generator fooling the discriminator with very poor-quality images because the discriminator's turn encourages good quality images. Theoretically, an equilibrium should be reached where the fake and real images are indistinguishable, and the generator can fool the discriminator 50% of the time.

The fairness loss is evaluated by creating another random binary label tensor `targets_outcome`. Unlike for the discriminator, we want to penalise the generator if the discriminator can predict the correct outcome label from the image. How we implement this depends on which fairness metric we are trying to minimise. We refer to Beutel *et al.* on how to build demographic parity and equality of opportunity into the loss functions[8]. The weighted combination between the two loss functions in the Combined Fairness GANs is novel to this work.

For demographic parity, we do not modify the terms in the prediction tensor `G_outcome` from the discriminator and the target label tensor `targets_outcome`. This decision means that the generator is penalised for all instances of the prediction being correct. Given that the discriminator is learning to predict the correct outcome label from real and fake images, this should mean that at equilibrium the discriminator assigns an outcome value with a 50% likelihood of being correct to each image, i.e. no better than random. Theoretically, this makes the outcome label y independent from the image X.

Since both classes (men and women) in the newly generated dataset will be assigned an attractiveness

label at random, the generated dataset should have a low (good) demographic parity score. There will be no reason for men to have a higher average score than women or vice versa. When a classifier is trained on this dataset and used to evaluate the real test set, it will steer the classifier into having less of a gap between the positive prediction distribution for men and women when tested on real images. Hence, the demographic parity score of the real data should also be lower.

The downside of this approach is that it might come at the cost of accuracy or other fairness metrics (for example, substantially lowering the positive prediction distribution for one subgroup). We will look at this in detail when evaluating the results.

```
def eo_loss(outputs, targets):

  l = []

  for i in range(GAN_BATCH_SIZE):
    if targets[i] == 0.0:
      l.append(i)

  index = torch.tensor(l)

  fair_outputs = outputs.index_select(0, index)
  fair_targets = targets.index_select(0, index)

  return fair_outputs, fair_targets
```

For equality of opportunity, we only count the loss for the positive terms in `targets_outcome`. We create a clone of `G_outcome` and iterate through the terms in the tensor `targets_outcome`. If the term is equal to 0 (we flip labels due to the heuristic used for the generator loss; hence we are looking for 0s and not 1s), we append this index to a list. At the end of the batch, we select only these indices from both tensors. This technique should induce the same random assignation of outcome values as before, but only to the images with 'ground truth' positive outcomes. The idea is to obtain demographic parity, but only within the positive outcome. This is the same as having equal true positive rates, which is equality of opportunity.

```
def cf_loss(outputs, targets):

  l = []

  for i in range(GAN_BATCH_SIZE):
    if targets[i] == 0.0:
      l.append(i)
    else:
      r = random.random()
      if r >= ALPHA:
        l.append(i)

  index = torch.tensor(l)

  fair_outputs = outputs.index_select(0, index)
  fair_targets = targets.index_select(0, index)

  return fair_outputs, fair_targets
```

For combined fairness, we observe that we make no adjustments to `G_outcome` `targets_outcome` and for demographic parity, but we only select the indices of `G_outcome` and `targets_outcome` when the values of `targets_outcome` are 0 for equality of opportunity. We can easily parameterise these two outcomes. We proceed as with equality of opportunity, but we also generate a random number between 0 and 1

each time we encounter a 1 in `targets_outcome`. If the generated number is greater than or equal to a parameter α, we preserve this index in the two tensors as well. The lower the value of α, the more likely it is that negative terms will be included in the loss, and the closer the loss is to the demographic parity loss. The higher the value of α, the closer the loss is to the equality of opportunity loss. Experiments were run for $\alpha \in [0.25, 0.5, 0.75]$.

3.8 Data input

```
class MyDataset(Dataset):
  def __init__(self, csv, root, size):
    self.f = pd.read_csv(csv, header=None)
    self.root = root
    self.size = size

  def __len__(self):
    return len(self.f)
```

The SCUT-FBP-5500 dataset came with a CSV file with two columns: the first was the image's filename, and the second was the average of the attractiveness ratings for the corresponding image. Pandas is used to read the CSV file into a dataframe for easy manipulation.

```
  def __getitem__(self, index):
    # generate correct file path
    img_path = os.path.join(self.root, self.f.iloc[index, 0])
```

The directory root is passed in as a parameter and is joined with the image file name stored in the first column and the ith row of the dataframe (where i = index).

```
    img = Image.open(img_path)
    img = img.resize((self.size, self.size), 0)
```

The Pillow library uses the image's file path to convert the JPEG image into the PIL format. PIL can easily be converted to and from PyTorch tensors. The image is then resized to the specified parameters. Our images were resized to 64 x 64 as that is the input size for the standard DCGAN architecture. Image sizes larger than this tend to need more complicated GAN architectures using upsampling and ResNets.

```
    edit = transforms.Compose([
      transforms.ToTensor(),
      transforms.Normalize([0.5], [0.5])])

    X = edit(img)
```

The PyTorch transforms module is used here to convert the PIL image into a tensor. Note that while PIL stores image values from 0 to 255, ToTensor divides all values by 255, resulting in values from 0 to 1. However, as we will use tanh activation in our generator, we need to resize the image values again to [-1, 1]. The resizing is done by normalising with a mean and variance (applied across all image channels) of 0.5.

```
    y = torch.zeros(1)
    if self.f.iloc[index, 1] >= BOUNDARY:
      y = torch.ones(1)

    s = torch.zeros(1)
    if self.f.iloc[index, 0][1] == 'F':
      s = torch.ones(1)

    return X, y, s
```

Finally, the values for y (outcome) and s (class) are created. These should technically be discrete variables but are treated as continuous in order for backpropagation to work. The ith entry at the second column (where i = index) is selected. This entry is the average attractiveness rating and is a real number from 0 to 5. However, thresholding attractiveness at 2.5 resulted in the dataset being very unbalanced, with 75.5% of people considered as attractive. Imbalanced classification is a different problem and often requires advanced normalisation techniques. Given that the point at which we threshold attractive is arbitrary (i.e. this report is not about who is attractive 'enough' to be labelled as such), we can change the boundary to achieve a balanced dataset. Through trial-and-improvement, we found that a boundary of 2.85 resulted in a total positive rate of 49.8%.

3.9 Early stopping

We are omitting the `Classifier` class here. It is identical to the Discriminator class, except that the output from the convolutional section of the model is only passed to one linear section instead of three.

The function for training the classifier is likewise mostly omitted: it follows a similar pattern to the `train_gan` function, except there is only one output from the classifier (its predictions about the labels) and only one component to the loss function (the difference between those predictions and the ground truth labels).

The relevant section of `train_classifier` is the implementation of early stopping.

```
while count < 5:

...

  if epoch % 3 == 0:

    _, val_acc = test_classifier(C, validation_loader)

    if val_acc > best_acc:
      count = 0
      best_acc = val_acc
      best_epoch = epoch
      save_classifier(C, text, CT.j, best_epoch)
    else:
      count += 1
```

Every third epoch, the validation set is passed into the `test_classifier` function, which returns the (overall procedure) accuracy of the classifier on the validation set. Since the validation set is not being used to train the classifier, we can stop the training when the validation set's accuracy stops rising and avoid overfitting to the train set, which would give poorer results on the test set. This procedure is known as early stopping.

However, we should also account for fluctuations in the validation set accuracy as the classifier trains. Therefore, we do not stop the first time that the validation accuracy is lower than before. Instead, we increment a `count` variable and stop if the validation accuracy is tested five times without beating its highest score. The value of 5 is a hyperparameter and will vary depending on how fast or slow a classifier trains.

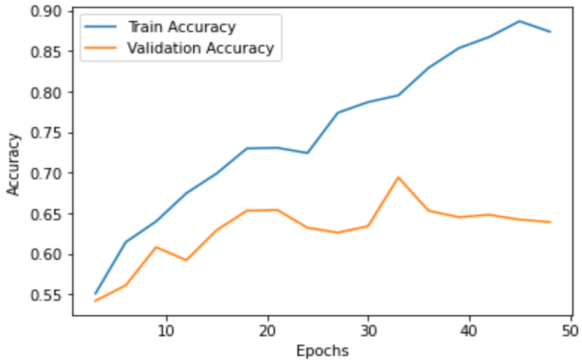


Figure 3.4: Classifier train and validation accuracy

This example stopped 15 epochs (5 checks) after the highest validation accuracy at 33 epochs. Each time a new highest validation accuracy was achieved, the weights of the classifier were automatically saved. The weights for the epoch with the highest validation accuracy were reloaded and used to classify the test set.

3.10 Training pipeline

Train/Validation/Test Split	7:2:2
GAN batch size	4
GAN epochs	400
Generator learning rate	0.0001
Discriminator learning rate	0.0001
Dimensions of generator noise	(4, 128, 1, 1)
Classifier learning rate	0.0005
Classifier batch size	8

Table 3.1: Hyperparameters

The real data was randomly and shuffled and split into a train set, a test set and a validation set. We repeated this three times to offset any statistical biases the random sampling process may have incurred. The real data set has 5500 images: for convenience of interpreting results, we split this in a 7:2:2 ratio: the train set had 3500 images, and the validation and test sets had 1000 each. These splits were saved to file so that they could be re-used after a notebook had been shut down.

Each type of GAN was trained (DP, EO, CF 0.5, CF 0.25, CF 0.75) on each of the three real train sets. The number of epochs was chosen as 400 after visual inspection of the pictures produced by the GANs. If time had allowed, it would have been preferable to use a metric like the Fréchet Inception Distance to calculate the epoch that produced the images with the highest fidelity (image quality) and diversity (number of different images that can be generated).

The trained generator was then used in evaluation mode (not updating its gradients) to generate a synthetic dataset of 3500 images. There were no constraints on generating the synthetic data: we did not generate a class term, and it was thought that balancing the outcomes might adversely affect the fairness implementation.

The following code illustrates how a synthetic dataset was generated from a trained GAN. TRAIN is a global variable set to 3500.

```
def generate_dataset(G, D, text, index):

  i = 0

  G.eval()
  D.eval()

  with torch.no_grad():

    while i < TRAIN:

      X = G.forward(torch.randn(GAN_BATCH_SIZE, NOISE, 1, 1)

      _, y = D.forward(X.detach())

      for j in range(GAN_BATCH_SIZE):

        if i < TRAIN:

          img = X[j].unsqueeze(0)
          i += 1
          if y[j].item() >= 0.5:
            z = 1
          else:
            z = 0
          save_data(img, z, -1, text, index, i)

  G.train()
  D.train()
```

The synthetic data was used to train a binary classifier, with the real validation set used to prevent overfitting via early stopping. Each classifier was trained ten times, with the classification results averaged. Finally, the averaged classification results for each type of GAN trained on the three different train set splits were averaged, leading to a total of 3 x 10 = 30 trials per type of GAN.

When testing the classifier, a function calculated whether the outcome value $\hat{y} \in \{0, 1\}$[2] predicted by the classifier was the same as the ground truth value $y \in \{0, 1\}$ of the real test set split. It used this and the class labels of the real test set to increment the appropriate Python dictionary entry for male true positive, female true positive, and so forth.

The dictionary was then passed into other functions to calculate the overall procedure accuracy, the positive prediction distribution and the true positive rate for each subgroup. The demographic parity and equality of opportunity scores were calculated, and a summary of the results was printed out to the console. These were copied and pasted into a Google Sheets spreadsheet (see Appendix 1). The average of the DP and EO scores for each type of GAN was also calculated.

The learning rates for the generator, discriminator and classifier, and the GAN and classifier's batch sizes were selected after an informal and inexhaustive trial-and-improvement style parameter sweep. This was necessary to save time, but it might not have been the most optimal configuration. In particular, the batch sizes are small. This choice made the models train faster but perhaps came at the cost of a more accurate calculation of the error gradient.

[2]The classifier outputted values between 0 and 1. Values less than 0.5 were rounded down to 0, while values greater than or equal to 0.5 were rounded up to 1.

Chapter 4

Results and Evaluation

We will begin by examining some statistics about the SCUT-FBP-5500 dataset to understand better what we are working with.

Statistic	Number	Percentage
Female Positives	1569	28.5%
Female Negatives	1181	21.5%
Male Positives	1171	21.3%
Male Negatives	1579	28.7%
Total Positives	2740	49.8%
Total Negatives	2760	50.2%
Total Females	2750	50.0%
Total Males	2750	50.0%
Total	5500	100.0%

Table 4.1: Dataset statistics for the real data

As mentioned earlier, the SCUT-FBP-5500 dataset is composed of 5500 images. We explained how we selected sex as the protected subgroup as there are 2750 images of male faces and 2750 images of female faces. Notice that a significantly higher number of women than men are considered attractive: 1569 women (28.5% of the whole dataset and 57.1% of women) compared to just 1171 men (21.3% of the whole dataset and 42.6% of men). Although it is atypical in cases of sex discrimination, we will thus be treating women as the privileged subgroup S_1 and men as the less-privileged subgroup S_0.

Recall also that the threshold value binarising the attractiveness ratings was chosen so that the ratio of total positives and total negatives was as close to 1:1 as possible. Although there is a slight imbalance (2740 positives to 2760 negatives), this is so tiny that it should not impact the classification.

We now test our hypothesis against the results obtained from the classifiers trained using data generated from different Combined Fairness GANs.

4.1 Testing the hypothesis

Five different types of GANs were trained in total:

1. DP: A Combined Fairness GAN with a loss function optimising for demographic parity.
2. EO: A Combined Fairness GAN with a loss function optimising for equality of opportunity.
3. CF 0.25: A Combined Fairness GAN with a loss function optimising for a weighted balance of demographic parity and equality of opportunity with the weighting parameter α set to 0.25.
4. CF 0.5: A Combined Fairness GAN with a loss function optimising for a weighted balance of demographic parity and equality of opportunity with the weighting parameter α set to 0.5.
5. CF 0.75: A Combined Fairness GAN with a loss function optimising for a weighted balance of demographic parity and equality of opportunity with the weighting parameter α set to 0.75.

As α represents the proportion of flipped positive target values that get ignored during the loss calculation, $\alpha = 0.5$ should, in theory, be halfway between equality of opportunity and demographic parity. $\alpha = 0.25$ should be closer to demographic parity and $\alpha = 0.75$ should be closer to equality of opportunity. These values were chosen to test the ability of the Combined Fairness GANs to parameterise the weighting between the two different types of unfairness.

Metric	**Real**	**DP**	**EO**	$\frac{\textbf{DP + EO}}{\textbf{2}}$	**CF 0.25**	**CF 0.5**	**CF 0.75**
Accuracy	0.763	0.663	0.615	0.639	0.643	0.626	0.666
DP Score	0.185	0.136	0.101	0.119	0.127	0.084	0.098
EO Score	0.142	0.128	0.11	0.119	0.112	0.076	0.063
Sum of DP and EO	0.327	0.264	0.211	0.238	0.239	0.16	0.161

Table 4.2: Summary of findings for the Combined Fairness GANs

This table shows the overall procedure accuracy, the demographic parity score, the equality of opportunity score and the sum of the demographic parity and equality of opportunity scores for the average of 30 classifiers trained on each type of GAN (split into three randomly seeded GANs with ten classifiers trained on each). It also shows these scores for the average of 30 classifiers trained on the real data (three random splits of the 5500 images into a 7:2:2 train/validation/test ratio and ten classifiers trained on each split). Finally, it shows the average of the scores achieved by the DP and EO Combined Fairness GANs.

Let us go through each of the terms of the hypothesis in Chapter 3 and see whether our results disprove the null hypothesis or not. We will calculate the results separately for each of CF 0.25, CF 0.5 and CF 0.75.

$$CF_{DP} + CF_{EO} < \frac{1}{2}(DP_{DP} + DP_{EO} + EO_{DP} + EO_{EO}) \tag{4.1}$$

$\frac{DP+EO}{2}$ Sum of DP score and EO score = 0.238

For CF 0.25:
Sum of DP score and EO score = 0.239
$0.239 > 0.238$, so the inequality does not hold for CF 0.25.

For CF 0.5:
Sum of DP score and EO score = 0.16
$0.16 < 0.238$, so the inequality holds for CF 0.5.

For CF 0.75:
Sum of DP score and EO score = 0.161
$0.161 < 0.238$, so the inequality holds for CF 0.75.

$$R_{DP} > CF_{DP} > DP_{DP} \tag{4.2}$$

$R_{DP} = 0.185$
$DP_{DP} = 0.136$

$CF\ 0.25_{DP} = 0.127$
$CF\ 0.5_{DP} = 0.084$
$CF\ 0.75_{DP} = 0.098$

All three of these values are lower than both R_{DP} and DP_{DP}. Technically this means the inequality does not hold, but this is more a flaw in forethought than a poor result. The fact the demographic parity score is lower in all the parameterised Combined Fairness GANs is potentially an exciting emergent behaviour.

$$R_{EO} > CF_{EO} > EO_{EO} \tag{4.3}$$

$R_{EO} = 0.142$
$EO_{EO} = 0.11$

$CF\ 0.25_{EO} = 0.112$
$CF\ 0.5_{EO} = 0.076$
$CF\ 0.75_{EO} = 0.063$

Again, the values for CF 0.5 and CF 0.75 are lower than both R_{EO} and EO_{EO}. The value for CF 0.25 is fractionally higher than EO_{EO} and much lower than R_{EO}, so the inequality holds in this case.

$$A_R - \epsilon = A_{DP} \approx A_{EO} = A_{CF} + \delta \tag{4.4}$$

$A_R = 0.763$
$A_{DP} = 0.663$
$A_{EO} = 0.615$

$A_{CF\ 0.25} = 0.643$
$A_{CF\ 0.5} = 0.626$
$A_{CF\ 0.75} = 0.666$

The gap in accuracy between the classifiers trained on the real data and the classifiers trained on generated data is stark: from 10 to 15 percentage points. The accuracy of the datasets generated by the various Combined Fairness GANs is reasonably consistent. However, the GAN optimised for demographic parity has noticeably higher accuracy than the GAN optimised for equality of opportunity (66.3% versus 61.5%). However, the differences between the generated dataset are not significant enough to disprove the second part of this equation. However, it is questionable whether 10-15 percentage points is an acceptably small value for epsilon.

$$L_{CF} = \alpha(L_{DP}) + \beta(L_{EO}) \tag{4.5}$$

We have shown in Chapter 3 how the combined fairness loss function functions on a continuum parameterised linearly by the parameter α. This could easily be rewritten in the form above.

In conclusion, inequality 4.1 holds for CF 0.5 and 0.75 and only narrowly misses being true for CF 0.25. For inequalities 4.2 and 4.3, it was assumed that the demographic parity score of the GAN optimised solely for demographic parity and the equality of opportunity score of the GAN optimised solely for equality of opportunity would be the lowest. The fact that this is not true does not invalidate the real meaning of the hypothesis, even though it does violate the specific inequality chosen. We have demonstrated that equation 4.5 is correct. In sum, the correctness of the hypothesis (for CF 0.5 and CF 0.75 at least) hinges on if we consider the drop in accuracy for the classifiers trained on the generated data to be too large for this method to be helpful.

4.2 Weighted combined fairness

The next item to examine is beyond the scope of the original hypothesis. After training a series of weighted demographic parity and equality of opportunity Combined Fairness GANs with $\alpha = 0.5$, the logical progression was to test GANs with $\alpha = 0.25$ and $\alpha = 0.75$ to see whether there were any obvious patterns in the transition from optimising for demographic parity to optimising for equality of opportunity and vice versa. Crucially, we can regard the Combined Fairness GAN optimised for demographic parity as CF 0: if $\alpha = 0$, all flipped positive terms get included in the outcome loss. Similarly, the Combined Fairness GAN optimised for equality of opportunity can be treated as CF 1: if $\alpha = 1$, no flipped positive terms get included in the outcome loss.

We produce an extended and rearranged version of the results table here in order to investigate the behaviour of the weighted Combined Fairness GANs.

Metric	**Real**	**DP (CF 0)**	**CF 0.25**	**CF 0.5**	**CF 0.75**	**EO (CF 1)**
Female TP	228.9	170	170.4	133.8	159	127.2
Female FP	60.4	56.8	68.6	47	52.1	44.6
Female TN	162	165.5	153.8	175.3	170.2	177.8
Female FN	54.4	113.4	112.9	149.5	124.4	156.1
Male TP	140.1	99.1	101.8	84.9	104.8	71.1
Male FP	51.4	55.1	67.2	52.4	52.2	45.1
Male TN	232.6	228.9	216.8	231.6	231.8	238.9
Male FN	70.3	111.3	108.5	125.4	105.5	139.3
Accuracy	0.763	0.663	0.643	0.626	0.666	0.615
Total PPD	0.48	0.38	0.407	0.318	0.367	0.288
Female PPD	0.572	0.448	0.47	0.359	0.416	0.338
Male PPD	0.387	0.312	0.344	0.276	0.318	0.237
DP Score	0.185	0.136	0.127	0.084	0.098	0.101
Total TPR	0.737	0.534	0.541	0.438	0.528	0.391
Female TPR	0.808	0.598	0.597	0.475	0.559	0.446
Male TPR	0.666	0.47	0.485	0.4	0.496	0.336
EO Score	0.142	0.128	0.112	0.076	0.063	0.11
Sum of DP + EO	0.164	0.132	0.12	0.08	0.081	0.106

Table 4.3: Full results for the Combined Fairness GANs

Before calculating the results, it was expected that either there would be no trend as we move from CF 0 (DP) to CF 1 (EO), or there would be a linear increase or decrease in the values echoing the linear weighting of the combined fairness loss. However, the generator and discriminator are non-linear, and it may be this that has led to the **emergent behaviour** in the demographic parity and equality of opportunity scores. As touched on in the last section, the lowest (and best) value for the DP score is not the Combined Fairness GAN optimised solely for demographic parity (DP / CF 0). In fact, this GAN has the highest demographic parity score except for the classifier trained on the real dataset. The CF 0.5 GAN has the best demographic parity score, followed by the CF 0.75 GAN. Given that the CF 0.25 GAN is the closest in weight to the DP (CF 0) GAN, it is surprising that it has a relatively high

demographic parity score. This may indicate that the decorrelation approach explained earlier may be flawed or incorrectly implemented.

The EO (CF 1) GAN does have a lower equality of opportunity score than the DP (CF 0) GAN, but both are eclipsed again by the CF 0.75 GAN, followed by the CF 0.5 GAN. It might be that adding a bit of noise by allowing a small fraction of the flipped positive labels to be used in the generator's outcome loss aids the decorrelation of the (unflipped) positive labels with the images, which decreases the equality of opportunity score. However, this would have to be tested extensively before a conclusion could be drawn.

The emergent behaviour demonstrated in the CF 0.75 GAN for the equality of opportunity and demographic parity scores can not be accounted for by a corresponding decrease in accuracy, as this GAN has the highest average downstream classification accuracy of all the different types of GANs. As the CF 0.5 GAN has the second-lowest average downstream classification accuracy, it cannot be ruled out that this accounts for at least some of the emergent behaviour in this case. However, this would need to be explored in further work.

This emergent behaviour is highly significant, as it demonstrates that at least CF 0.75 (and possibly CF 0.5) would perform better regarding the equality of opportunity and demographic parity scores than an ensemble model composed of a DP (CF 0) GAN and an EO (CF 1) GAN. It would also use half the parameters and train in half the time. This result demonstrates the true potential and versatility of the Combined Fairness GAN.

Finally, we might ask if the emergent behaviour is counterbalanced by a trade-off elsewhere in the data. The answer is unless the GAN has been structurally improved by the weighted fairness loss, then there must be a trade-off somewhere. However, it could be a worthwhile trade-off if we are more interested in demographic parity or equality of opportunity that the metric or metrics that have gotten worse. On the other hand, we have seen that there are many potential ways to evaluate fairness, and we have not had the time to compare them all.

As shown below, there is no significant trade-off in the overall true negative rate or the difference between the female and male true negative rates in CF 0.5 and CF 0.75 compared to DP and EO. There appears to be a slight decrease of the female overall procedural accuracy in CF 0.5, which once more suggests that CF 0.75 might be a better display of emergent behaviour. However, the female overall procedural accuracy in CF 0.5 is still higher than in DP.

Metric	Real	DP	EO	$\frac{\text{DP} + \text{EO}}{2}$	CF 0.25	CF 0.5	CF 0.75
Female TNR	0.728	0.744	0.799	0.772	0.692	0.789	0.766
Male TNR	0.819	0.806	0.841	0.824	0.763	0.815	0.816
TNR Difference	-0.091	-0.062	-0.042	-0.052	-0.071	-0.026	-0.05
Overall TNR	0.779	0.779	0.823	0.801	0.732	0.804	0.794

Table 4.4: True negative rates not significantly affected

Metric	Real	DP	EO	$\frac{\text{DP} + \text{EO}}{2}$	CF 0.25	CF 0.5	CF 0.75
Female Accuracy	0.773	0.663	0.603	0.633	0.64	0.612	0.651
Male Accuracy	0.754	0.664	0.627	0.646	0.645	0.641	0.681
Accuracy Difference	0.019	-0.001	-0.024	-0.013	-0.004	-0.029	-0.03
Overall Accuracy	0.763	0.663	0.615	0.639	0.643	0.626	0.666

Table 4.5: Female overall procedural accuracy for CF 0.5 is slightly low

There are two more interesting things to note from the results above. Firstly, we noted earlier that the weighted Combined Fairness GANs might work by increasing the number of false positives for women (the privileged subgroup in our example) when shifting from a low equality of opportunity score to a low demographic parity score. The theory behind this was that as the privileged subgroup, women would have a higher percentage of true positives to start off with and the classifier would therefore need to boost the number of false positives in order to improve the female positive predictive distribution. Men do seem to have a lower true positive rate than women (despite only a very slight difference in the outcome

distribution) and we can see that the female false positive rate is indeed highest in CF 0.25, which is the closest weighted Combined Fairness GAN to DP. However, the fact that the false positive rate is lower in DP itself may mean that this observation is unrelated.

Secondly, the values for the total positive predictive distribution and especially the total true positive rate are much lower in the classifiers trained on generated data than the classifiers trained on real data. For an allocative decision-making process, this means that people (qualified people in the case of the true positive rate) miss out on opportunities or resources. It seems like the classifiers trained on the generated data are attaining lower equality of opportunity and demographic parity scores by decreasing the positive predictive distributions and the true positive rates for both women and men. This is a potential drawback to this approach and would have to be considered in every decision-making context where Combined Fairness GANs might be applicable.

4.3 Comparison to Fairness GAN

As we have already discussed, another possible weakness of this work is that we don't have results on the same dataset using the same or a similar technique to compare it to. If this work were to be extended, we would definitely have to repeat the method with other datasets that have known results.

Here we briefly compare our results to that of Sattigeri *et al.*'s Fairness GAN, noting again that they use different datasets, a more complicated architecture and a different way of decorrelating the outcome from the protected attribute (their method is direct whereas the method in this work indirectly produces the decorrelation between the outcome and the protected attribute by adjusting how the outcome is assigned to each image).

Sattigeri *et al.* report the error rate for their results, which is just 1 minus the overall procedural accuracy. Surprisingly (unless I am misinterpreting the data), the error rates they obtained with their DP and EO Fairness GANs on their three datasets were quite high: from a low of 0.0890 on the Quick, Draw! dataset with their EO Fairness GAN right up to 0.5118 on the Soccer dataset with their DP GAN, which is worse than random. Their lowest demographic parity score is 0.0096 on the CelebA dataset with their EO Fairness GAN - it appears that their GANs also didn't always correlate neatly with their intended purpose. This is almost ten times lower than our lowest demographic parity score of 0.084 with the CF 0.5 Combined Fairness GAN. Their error rate in this case was 0.3238, making their downstream classification more accurate than CF 0.5 (0.626).

Their lowest equality of opportunity score is 0.0160 on the Soccer dataset with their EO GAN. Interestingly, their EO GANs seem to produce a lower equality of opportunity score than their DP GANs, but the opposite is not always true. We have observed exactly the same in our models. We speculate that subgroups that already have similar true positive rates could be more likely to have similar positive predictive distributions than the other way around, because positive predictive values can consist of false positives and true positives whereas the true positive rate is (obviously) restricted to true positives. Again, this is much lower than our lowest equality of opportunity score. However, this is another result with an accuracy of less than 0.5, demonstrating once more the trade-off between accuracy and fairness.

4.4 Discussion of low classification accuracy

First, we must consider that the classification accuracy for the real test set was low to start with, with an average of just 76.3%. One of the weaknesses of the current work is that insufficient time was available to adjust the hyperparameters and architecture of the classifier in order to improve its accuracy. Some changes were tested, but none yielded better results. Re-using the DCGAN discriminator architecture for the classifier was likely the wrong decision, and an alternative design could have been sought and implemented. A drop of 10 percentage points might have been more of a reasonable trade-off if the classification accuracy on the real test set was 95% or above.

Moreover, part of the accuracy gap may stem from the quality and diversity of the generated data. While the GAN managed to generate reasonably good quality images (see Appendix 1 for examples), they are not as realistic as the real data. Feature extraction through convolutional layers has resulted in the same small-ish set of features repeatedly appearing in slightly different combinations. This issue is different and more intractable than mode collapse. However, it shares the same result: a decrease in the diversity of the images generated (the latent space of the train set's distribution is not fully covered). Given the binary classifier was trained on the generated data but tested on the real data, it is possible that sub-optimal generated images led to classification weights being learned that did not transfer well to the test set.

While decorrelating the outcome from the image appeared to work well in lowering the demographic parity and equality of opportunity scores across all the classifiers trained on generated data, the very act of decorrelation may have made the links between the generated images and the generated labels harder to learn, as the connections between specific arrangements of facial features and attractiveness would have been weaker.

It is also plausible that the gap between the classification accuracy for the real data and the synthetic data might widen if the classification accuracy for the real data was improved. The more accurate the classifier, the fewer false positives and false negatives it can adjust to meet a fairness criterion. Therefore, it might meet that fairness criterion by turning true positives into false negatives or true negatives into false positives (or both). It is questionable whether this would be fair on the individuals affected by these misclassifications, even if it results in a fairer score for the subgroup they belong to. The notion of individual fairness was perhaps dismissed a little too hastily at the start, as it is essential to consider both the individual and the group(s) they belong to when evaluating a binary classifier's results.

We must not get too hung up on the low classification accuracies without considering the bigger picture. Since the whole premise is that the initial dataset contains bias, some of the labels must be incorrect. From this perspective, high accuracy is correlated with a faithful reproduction of existing patterns of discrimination. Although we train our binary classifiers on debiased generated data, we use the real test set for evaluation (or else the results would have no external validity). We also use the real train set in the very act of generating unbiased data. Hence the classification results still contain echoes of the human decision making embedded in the original dataset, even if they are fairer on paper. They could also be fairer in reality, but we can never observe reality without leaving the faint traces of measurement errors.

We do not and possibly cannot know which labels are incorrect and how much accuracy is misleading. As Mowbray states, '[i]t is a policy choice whether or not to accept reduced accuracy, at least in the short term, in return for meeting fairness conditions'[23].

We must also remember that the impact of the same numerical amount of unfairness may be different for different subgroups, which might influence the strength of the views they hold about different outcomes. The method of fairness weighting developed in this work would help to readdress these discrepancies if the parties involved agreed to shift further to the preference of the disadvantaged subgroup than the raw numbers would suggest. Without considering these externalities, it is easy to dismiss another subgroup's perspective as irrational and make decisions solely based on your idea of what type of fairness would most benefit somebody else.

A related idea is that weighted fairness could express doubt or uncertainty about the consequences of a decision-making process. If it is not known in advance whether false positives are more costly than false negatives, or if the decision is representative or allocative (or a combination), then balancing two or more different fairness metrics might be a safe way to hedge your bets. A weighted measure of fairness is also a weighted measure of unfairness, and under uncertain conditions, it might be a better approach to minimise unfairness than to maximise fairness.

4.5 Learning curves

We can see that the Combined Fairness GANs did not reach equilibrium by graphing the discriminator and the generator losses, in addition to the discriminator's accuracy (which measures how good the discriminator is at telling whether an image is real or fake and, by extension, how good the generator is at generating images that are realistic enough to fool the discriminator). These examples are taken from a CF 0.5 GAN trained for 400 epochs, but similar results were found for all the types of Combined Fairness GAN.

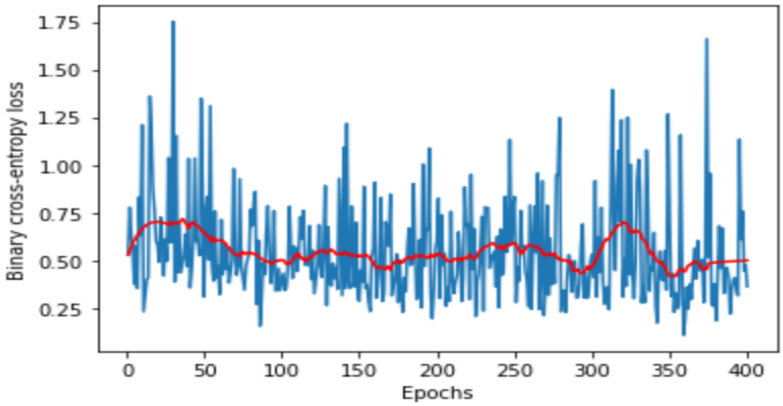


Figure 4.1: Example of discriminator loss

The raw loss at the end of each epoch is blue. (The loss is actually calculated every batch but the graph is unreadable unless the loss is taken less frequently.) The red line is a smoothed version of the loss using the Savitsky-Golay filter, which removes some of the signal noise.

Because a GAN is a mini-max adversarial game that does not converge but instead (theoretically) reaches an equilibrium between the two players, the loss functions are less interpretable than single-model losses, which trend down towards zero. The fact that the smoothed loss (red line) is fairly smooth and mostly stays within the range 0.5 to 0.75 indicates that there are no major errors. However, the consistently large oscillations that peak at a binary cross-entropy loss of up to 1.75 suggest that the training is functional but not optimal.

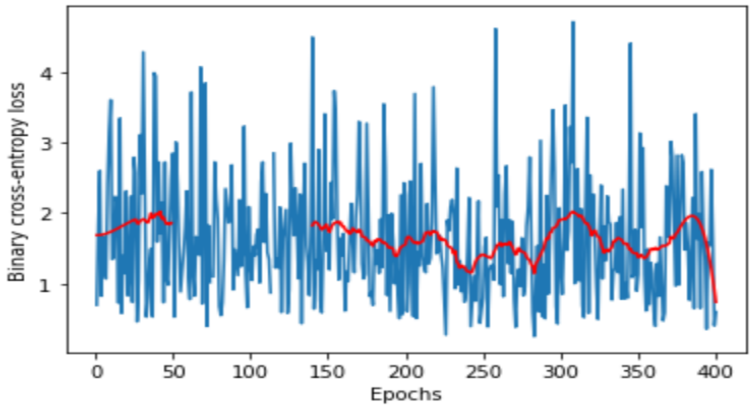


Figure 4.2: Example of generator loss

Similar oscillations are also present in the generator loss. The upper range of the oscillations for the generator loss is much higher than those for the discriminator loss, which might indicate that the generator is losing against the discriminator in the adversarial game. The Savitsky-Golay smoothed loss is also less stable (using the same parameters as for the discriminator loss). In order to connect the gap, you have to decrease the window size of the filter so much that the smoothed loss almost resembles the noisy signal.

There was not enough time to work out what this gap represents, and there are no obvious signs from the raw loss. It could be related to the rate of improvement of the generated images, which does slow markedly around epoch 150. However, it speeds up long before epoch 50, so this is probably not the case.

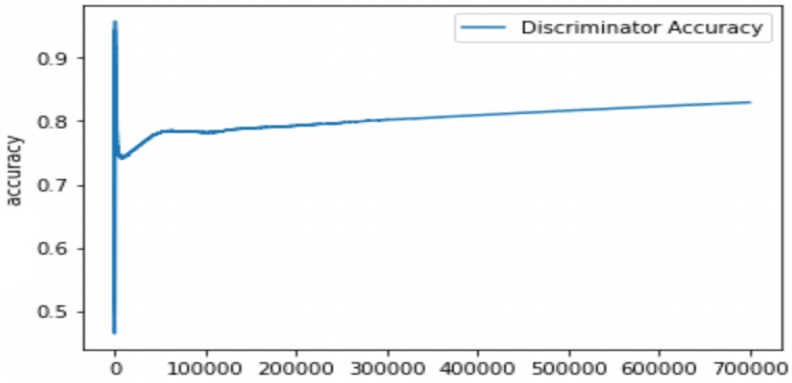


Figure 4.3: Example of discriminator accuracy

Initially, the discriminator makes random guesses at the image source, so the accuracy starts at 0.5 (or 50%). The accuracy quickly rises to close to 100% in the early stages of training as the generator produces very bad quality images. However, the generator soon manages to fool the discriminator around a quarter of the time. The images generated at this point start to resemble faces but still look very odd.

Note that the generated images are never good enough to fool the discriminator 50% of the time. This may explain why they never reach the quality of the real images. However, it is difficult to tell without further investigation whether the generator is too weak or the discriminator is too strong.

The discriminator gradually gets more accurate throughout the remainder of the training, even though the generated images continue to increase in quality. The training was stopped at 400 epochs because that was roughly when (on visible inspection at least) the image quality did not get noticeably better when trained for further epochs. This graph suggests that image quality should not have been the only factor considered when deciding to stop training. It would be interesting to see whether the images generated around batch 100,000 would have yielded better classification results despite being visually less appealing. The FID scores of the generated data at various training stages could also have been calculated and compared. Given that this work's primary purpose was to improve a binary classifier's fairness, not to produce high quality generated images, the assumption that higher quality images would benefit training ought to have been double-checked.

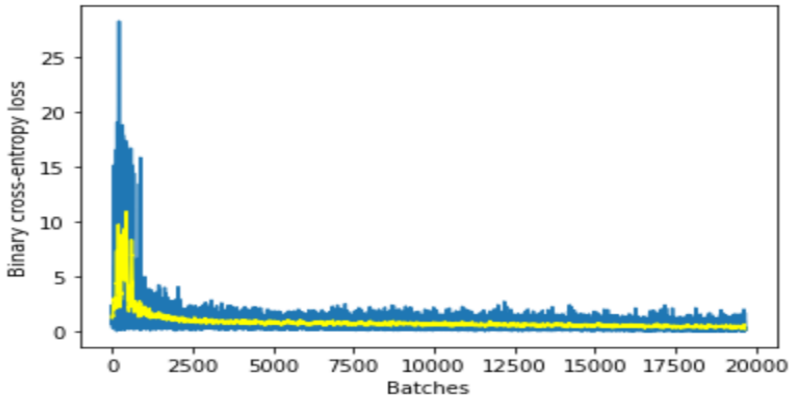


Figure 4.4: Example of classifier accuracy

In contrast to the discriminator and generator, the binary cross-entropy loss for the classifier exhibits the typical behaviour for a neural network loss function - a high drop in loss at the very start as the Adam optimiser makes larger adjustments to the weights, then a steady decrease towards a value close to zero as the momentum is decreased. This suggests that the model is performing at full capacity for these hyperparameters. This means either the model or the hyperparameters need to be changed (most likely both), but the classifier itself is training correctly.

Chapter 5

Conclusions

This work has successfully built on the results of Sattigeri *et al.* by successfully demonstrating emergent behaviour in weighted demographic parity and equality of opportunity Combined Fairness GANs. If you relax the hypothesis so that a ten percentage point loss in accuracy between the classifiers trained on the real data and the classifiers trained on the generated data is acceptable, then the null hypothesis can be rejected for the weighted Combined Fairness GANs with $alpha = 0.5$ and $alpha = 0.75$. This work has achieved its objective in demonstrating that a Combined Fairness GAN can be trained with a linear combination of a loss function optimised for demographic parity and a loss function optimised for equality of opportunity and generate data that, when used to train a binary classifier, produces a lower combined demographic parity and equality of opportunity score than the average of this combined score for a GAN optimised solely for demographic parity and a GAN optimised solely for equality of opportunity. In showing this is possible in high-dimensional image data, this work has made a valuable contribution to the field of fairness in algorithmic decision making.

The practical implications of improving the fairness of image classification are immense. Automated decision-making processes based on images are becoming increasingly prevalent in our society: two prominent examples are facial recognition and computer vision in driverless cars. While it is vital not to perpetuate or exacerbate harms caused by biases embedded in the data we use, it is not always feasible to collect new and unbiased data.

The method offered here also recognises the complexity of fairness and the diverse perspectives people have on what constitutes a fair decision. While the field has come to acknowledge that total fairness is impossible, there is a paucity of options that explicitly try to reach a compromise. If the conclusions of this paper hold for higher accuracy rates and across different domains, they will be invaluable for reorienting the perspective of the fairness community towards a multi-stakeholder, compromise-based approach.

All fairness measures are open to abuse, and we ought to respond with appropriate regulations or principles. That requires better communication to promote a greater non-technical understanding of these issues. We must harness multi-disciplinary expertise and integrate perspectives from all walks of life, making sure to include people outside of academia and C-suites. Policy-makers and technical experts need to work more closely together. Philosophers and ethicists are also well situated to examine the costs of different fairness trade-offs in algorithmic decision-making. We must also recognise and tackle ethics-washing and so-called 'fairness gerrymandering' by organisations wishing to avoid new regulations.

Combined fairness metrics like the ones discussed here have the most value in situations where different stakeholders have different priorities and attach different costs to certain types of unfairness. The less insular we are, the more we can discover a solution that works - at least partially - for all parties.

The top three hypotheses stemming from this project that could prompt further research are:

1. If the base classification accuracy (i.e. the accuracy of the binary classifier on the real data) is improved to industry standards, does the weighted fairness loss still function as well as it currently does or will it behave differently?

2. In this work, the dataset was carefully chosen so that the classes (male and female) were balanced. Since many protected subgroups are also minority subgroups, do the results still hold for imbalanced classification? If not, could additional techniques be developed to mitigate this?

3. Can the conclusions be extended to other common fairness metrics such as equality of odds? Could you make a three-way trade-off? An n-way trade-off?

There are many other directions future research could take. A more sophisticated architecture could be used for the GANs (e.g. ResNet), and a more systematic hyperparameter search could be conducted. The hypothesis needs to be tested on other datasets to have external validity. Progressive growing techniques could be used to gradually build up the size of the generated images, allowing us to reach larger image sizes that would be more relevant for real-world applications. The more complicated Fairness GAN loss function could be implemented. The source and outcome terms of the loss function might be weighted differently, or the terms could alternate for each batch of training. The method could be extended to an n-ary classifier, or fairness could be measured for more than two subgroups. Protected characteristics can overlap, and intersectional fairness would be interesting from a mathematical and a social standpoint. Finally, a formal proof of the functionality and limitations of the combined fairness method could be developed.

Appendix A

Comparison of generated faces

Figure A.1: Examples of faces classified as attractive using data generated by a DP GAN

Figure A.2: Examples of faces classified as not attractive using data generated by a DP GAN

Figure A.3: Examples of faces classified as attractive using data generated by a EO GAN

Figure A.4: Examples of faces classified as not attractive using data generated by a EO GAN

Figure A.5: Examples of faces classified as attractive using data generated by a CF 0.5 GAN

Figure A.6: Examples of faces classified as not attractive using data generated by a CF 0.5 GAN

Appendix B

Full results for all training splits

	Real Data				Generated Data Demographic Parity			
Results	**1**	**2**	**3**	**AVG**	**1**	**2**	**3**	**AVG**
Female TP	233	241.6	212.1	228.9	200.5	162.8	146.6	170
Female FP	56.6	64.5	60	60.4	57.1	54.8	58.6	56.8
Female TN	167.4	157.5	161	162	166.9	167.2	162.4	165.5
Female FN	58	46.4	58.9	54.4	90.5	125.2	124.4	113.4
Male TP	148.1	132	140.1	140.1	109.3	87.5	100.4	99.1
Male FP	50.1	44.1	60	51.4	56.4	44.5	64.4	55.1
Male TN	219.9	243.9	234	232.6	213.6	243.5	229.6	228.9
Male FN	66.9	70	73.9	70.3	105.7	114.5	113.6	111.3
Accuracy	**0.768**	**0.775**	**0.747**	**0.763**	**0.69**	**0.661**	**0.639**	**0.663**
Female PPD	0.563	0.6	0.553	0.572	0.5	0.427	0.417	0.448
Male PPD	0.409	0.359	0.394	0.387	0.342	0.269	0.324	0.312
DP Score	**0.154**	**0.241**	**0.159**	**0.185**	**0.159**	**0.157**	**0.093**	**0.136**
Female TPR	0.801	0.839	0.783	0.808	0.689	0.565	0.541	0.598
Male TPR	0.689	0.654	0.655	0.666	0.508	0.433	0.469	0.47
EO Score	**0.112**	**0.185**	**0.128**	**0.142**	**0.181**	**0.132**	**0.072**	**0.128**
Sum of DP and EO	**0.266**	**0.426**	**0.287**	**0.327**	**0.34**	**0.289**	**0.165**	**0.264**

	Generated Data Equality of Opportunity				Generated Data Combined Fairness Alpha = 0.25			
Results	**1**	**2**	**3**	**AVG**	**1**	**2**	**3**	**AVG**
Female TP	191.8	99.3	90.6	127.2	217.5	188.3	105.5	170.4
Female FP	76.5	31.2	26	44.6	91.9	71.5	42.3	68.6
Female TN	147.5	190.8	195	177.8	132.1	150.5	178.7	153.8
Female FN	99.2	188.7	180.4	156.1	73.5	99.7	165.5	112.9
Male TP	112.3	52.3	48.6	71.1	125.1	104.6	75.8	101.8
Male FP	83.7	30.3	21.4	45.1	83.3	69.3	49	67.2
Male TN	186.3	257.7	272.6	238.9	186.7	218.7	245	216.8
Male FN	102.7	149.7	165.4	139.3	89.9	97.4	138.2	108.5
Accuracy	**0.638**	**0.6**	**0.607**	**0.615**	**0.661**	**0.662**	**0.605**	**0.643**
Female PPD	0.521	0.256	0.237	0.338	0.601	0.51	0.3	0.47
Male PPD	0.404	0.169	0.138	0.237	0.43	0.355	0.246	0.344
DP Score	**0.117**	**0.087**	**0.099**	**0.101**	**0.171**	**0.155**	**0.055**	**0.127**
Female TPR	0.659	0.345	0.334	0.446	0.747	0.654	0.389	0.597
Male TPR	0.522	0.259	0.227	0.336	0.582	0.518	0.354	0.485
EO Score	**0.137**	**0.086**	**0.107**	**0.11**	**0.166**	**0.136**	**0.035**	**0.112**
Sum of DP and EO	**0.254**	**0.173**	**0.206**	**0.211**	**0.337**	**0.291**	**0.09**	**0.239**

	Generated Data Combined Fairness Alpha = 0.5				Generated Data Combined Fairness Alpha = 0.75			
Results	**1**	**2**	**3**	**AVG**	**1**	**2**	**3**	**AVG**
Female TP	148.7	84.3	168.5	133.8	215.2	126.6	135.1	159
Female FP	44.9	23.2	72.9	47	85.6	29.1	41.6	52.1
Female TN	179.1	198.8	148.1	175.3	138.4	192.9	179.4	170.2
Female FN	142.3	203.7	102.5	149.5	75.8	161.4	135.9	124.4
Male TP	88	38.5	128.2	84.9	136.9	76.9	100.7	104.8
Male FP	41	16.6	99.5	52.4	81.2	32.1	43.3	52.2
Male TN	229	271.4	194.5	231.6	188.8	255.9	250.7	231.8
Male FN	127	163.5	85.8	125.4	78.1	125.1	113.3	105.5
Accuracy	**0.645**	**0.593**	**0.639**	**0.626**	**0.679**	**0.652**	**0.666**	**0.666**
Female PPD	0.376	0.211	0.491	0.359	0.584	0.305	0.359	0.416
Male PPD	0.266	0.113	0.448	0.276	0.45	0.222	0.283	0.318
DP Score	**0.11**	**0.098**	**0.043**	**0.084**	**0.134**	**0.083**	**0.076**	**0.098**
Female TPR	0.511	0.293	0.622	0.475	0.74	0.439	0.499	0.559
Male TPR	0.409	0.191	0.599	0.4	0.637	0.381	0.471	0.496
EO Score	**0.102**	**0.102**	**0.023**	**0.076**	**0.103**	**0.059**	**0.028**	**0.063**
Sum of DP and EO	**0.212**	**0.2**	**0.066**	**0.16**	**0.237**	**0.142**	**0.104**	**0.161**

Appendix C

Code for an example training cycle

Train Combined Fairness (DP and EO) GAN

```
D_CF = Discriminator(NUM_FEATURES)
G_CF = Generator(NOISE, NUM_FEATURES)
T_CF = Tracker()

G_CF.to(device)
D_CF.to(device)

initialise_weights(D_CF)
initialise_weights(G_CF)
```

```
train_loader = read_split(f'Real/Splits/{SPLIT}', GAN_BATCH_SIZE, 'Train', load=True, fair=False)
```

```
%%time
ALPHA = 0.5
train_gan(G_CF, D_CF, T_CF, train_loader, 400, 'CF', 20)
```

```
plot_gan(T_CF)
```

```
generate_images(G_CF)
```

Load in GAN checkpoints

```
G_CF, D_CF, T_CF = load_gan('CF', 999, 400)
```

Generate data from trained GAN

```
generate_dataset(G_CF, D_CF, 'CF', f'Images/{SPLIT}')
```

```
X_CF, y_CF = read_tensors('CF', f'Images/{SPLIT}', TRAIN, fair=True)
cf_data = FairDataset(X_CF, y_CF)
```

Save and load splits

```
cf_train_loader = DataLoader(cf_data, CLASSIFIER_BATCH_SIZE, shuffle=True)
validation_loader = read_split(f'Real/Splits/{SPLIT}', CLASSIFIER_BATCH_SIZE, 'Validation', load=True, fair=False)
test_loader = read_split(f'Real/Splits/{SPLIT}', CLASSIFIER_BATCH_SIZE, 'Test', load=True, fair=False)
```

Test classifier using generated dataset

```
for i in range(10):
    C_CF = Classifier(NUM_FEATURES)
    CT_CF = C_Tracker()

    C_CF.to(device)

    best_epoch = train_classifier(C_CF, CT_CF, cf_train_loader, validation_loader, 'CF', fair=True)

    C_CF = load_classifier('CF', CT_CF.j, best_epoch)

    print(CT_CF.j)
    print(best_epoch)
    d = test_classifier(C_CF, test_loader)
    print_results(d, False)
    get_accuracy(d, False)
    get_dp(d, False)
    get_eo(d, False)
    print('---')
```

```
plot_classifier(CT_CF)
```

Bibliography

[1] Equality Act 2010, c.1. Available from https://www.legislation.gov.uk/ukpga/2010/15/part/2/chapter/1 [Accessed 12 April 2021].

[2] Tameem Adel, Isabel Valera, Zoubin Ghahramani, and Adrian Weller. One-network adversarial fairness. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 2412–2420, 2019.

[3] Micah Altman, Alexandra Wood, and Effy Vayena. A harm-reduction framework for algorithmic fairness. *IEEE Security & Privacy*, 16(3):34–45, 2018.

[4] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine Bias, 2016. ProPublica. Available from https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing [Accessed 12 April 2021].

[5] Hmrishav Bandyopadhyay. Understanding ACGANs with code. Available from https://towardsdatascience.com/understanding-acgans-with-code-pytorch-2de35e05d3e4, 2020. [Accessed 15 April 2021].

[6] Solon Barocas, Elizabeth Bradley, Vasant Honavar, and Foster Provost. Big data, data science, and civil rights. *arXiv preprint arXiv:1706.03102*, 2017.

[7] Richard Berk, Hoda Heidari, Shahin Jabbari, Michael Kearns, and Aaron Roth. Fairness in criminal justice risk assessments: The state of the art. *Sociological Methods & Research*, page 0049124118782533, 2018.

[8] Alex Beutel, Jilin Chen, Zhe Zhao, and Ed H Chi. Data decisions and theoretical implications when adversarially learning fair representations. *arXiv preprint arXiv:1707.00075*, 2017.

[9] François Chollet. *Deep Learning with Python*. Manning, 2017.

[10] Alexandra Chouldechova. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big data*, 5(2):153–163, 2017.

[11] Sam Corbett-Davies and Sharad Goel. The measure and mismeasure of fairness: A critical review of fair machine learning. *arXiv preprint arXiv:1808.00023*, 2018.

[12] Kate Crawford. The trouble with bias, 2017. NIPS keynote speech. Available from https://youtu.be/6Uao14eIyGcg [Accessed 14 April 2021]. Transcribed text from https://gizmodo.com/microsoft-researcher-details-real-world-dangers-of-algo-1821129334 [Accessed 14 April 2021].

[13] Paul Davidoff. Working toward redistributive justice. *Journal of the American Institute of Planners*, 41(5):317–318, 1975.

[14] Miguel Delgado-Rodriguez and Javier Llorca. Bias. *Journal of Epidemiology & Community Health*, 58(8):635–641, 2004.

[15] William Dieterich, Christina Mendoza, and Tim Brennan. Compas risk scales: Demonstrating accuracy equity and predictive parity. *Northpoint Inc*, 7(7.4):1, 2016.

[16] Ian Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[17] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

[18] Moritz Hardt, Eric Price, Nati Srebro, et al. Equality of opportunity in supervised learning. In *Advances in neural information processing systems*, pages 3315–3323, 2016.

[19] Brendan F Klare, Mark J Burge, Joshua C Klontz, Richard W Vorder Bruegge, and Anil K Jain. Face recognition performance: Role of demographic information. *IEEE Transactions on Information Forensics and Security*, 7(6):1789–1801, 2012.

[20] Lingyu Liang, Luojun Lin, Lianwen Jin, Duorui Xie, and Mengru Li. Scut-fbp5500: A diverse benchmark dataset for multi-paradigm facial beauty prediction. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 1598–1603. IEEE, 2018.

[21] George Loewenstein and Jennifer S Lerner. The role of affect in decision making. *Handbook of affective science*, 619(642):3, 2003.

[22] Takeru Miyato and Masanori Koyama. cgans with projection discriminator. *arXiv preprint arXiv:1802.05637*, 2018.

[23] Miranda Mowbray. Predictive analytics: you can't have it all. University of Bristol School of Mathematics Colloquium Series, 2018.

[24] Arvind Narayanan. Tutorial: 21 fairness definitions and their politics. ACM FAT* 2018. Available from https://youtu.be/jIXIuYdnyyk [Accessed 12 April 2021].

[25] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2642–2651. JMLR. org, 2017.

[26] Aladdin Persson. Machine Learning Collection. Available from https://github.com/aladdinpersson/Machine-Learning-Collection/tree/master/ML/Pytorch/GANs, commit: 42a8161013a8cbb2198ba47ac48d77f3e9127454, 2020.

[27] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[28] Dorothy Roberts. The social and moral cost of mass incarceration in african american communities. *Stanford Law Review*, 56:1271–1305, 2014.

[29] Lorna Roth. Looking at Shirley, the ultimate norm: Colour balance, image technologies, and cognitive equity. *Canadian Journal of Communication*, 34(1), 2009.

[30] Prasanna Sattigeri, Samuel C Hoffman, Vijil Chenthamarakshan, and Kush R Varshney. Fairness gan: Generating datasets with fairness properties using a generative adversarial network. *IBM Journal of Research and Development*, 63(4/5):3–1, 2019.

[31] Amarjot Singh. Children safety retrieval (censer) system for retrieval of kidnapped children from brothels in india. Available from https://tinyurl.com/censer-link, 2020. [Accessed 15 April 2021].

[32] Alex Stevens. *Drugs, crime and public health: The political economy of drug policy*. Routledge-Cavendish, 2010.

[33] Rashid Tariq. *Make Your First GAN With PyTorch*. 2020.

[34] Sahil Verma and Julia Rubin. Fairness definitions explained. In *2018 IEEE/ACM International Workshop on Software Fairness (FairWare)*, pages 1–7. IEEE, 2018.

[35] Depeng Xu, Shuhan Yuan, Lu Zhang, and Xintao Wu. Fairgan: Fairness-aware generative adversarial networks. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 570–575. IEEE, 2018.