Uncovering Bias in Image Classification Models Using Counterfactual Explanations

AMTH 491: Applied Mathematics Senior Project

Author: Anna Matthes
Advisers: Mariel Vasquez and Austin Narcomey
Yale University, December 2022

1. Introduction

With machine learning models being used more often in everyday life, the accuracy and understanding of these models is becoming more important to society as a whole. Without further exploration of a model, the reasons for the larger decisions of these models can be left unknown. Unwanted biases can be hidden within the decision making of a model and these biases can negatively affect those for whom the model's decisions are made. The consequences of having biases in these models could be very large, particularly as facial recognition is being used more often in security, (NEC EVA 2020), law enforcement (Goodwin and Wright, 2021), and healthcare (Crompton and Diamond, 2019). The stakes of the decisions these models are making are very high, so it is important to have methods to discover and measure the biases so that they can be appropriately mitigated or modified. These biases are also often targeted toward minority groups who already receive biased treatment in these areas (Garvie and Frankle 2016; Halamka et al. 2022), so finding and dealing with the bias of these models is an important step in solving unfair discrimination.

The method that we decided to use to uncover the biases in a model is using counterfactual explanations. Counterfactual explanations are a useful method to gain insight into

a model's decisions. A counterfactual is a controlled change to the input of the model. They can uncover the specifics of what factors determine a model's decision by observing how the output changes with the change of input (Verma et al. 2022). For our case, a counterfactual explanation can uncover bias within an image classification model by showing how a change in a demographic feature affects the output. The key to creating the image counterfactuals is finding a way to change only the desired attribute without affecting the rest of the image. If the rest of the image remains unchanged, it can be inferred that any changes in the output are caused only by the change in that attribute. By using a method called FaderNetworks (Lample et al. 2018) to create the counterfactuals, this project attempts to change only the sensitive attribute in a realistic and holistic way, while leaving the rest of the photo relatively unchanged. Once the attributes are changed, we are able to analyze the model's biases based on the classifications generated by the counterfactuals and the model.

2. Related work

Several recent papers have attempted to use image counterfactuals to find bias. They use methods other than FaderNetworks to generate their counterfactuals and use their own definitions of bias to analyze the classifications of their generated counterfactuals.

Dash et al. 2022 use a method called ImageCFGen to generate their counterfactuals. This method is based on a Deep Structural Causal Model and consists of an encoder-generator architecture. In their analysis of the classifications of their counterfactuals they focus only on quantifying the bias that their entire diverse set of counterfactuals have towards a single attribute, "attractive". This paper also provides a method of mitigating the bias that it measures.

Denton et al. 2020 use a similar method to FaderNetworks to generate their counterfactuals. They use a GAN along with training a linear classifier to discriminate between

latent codes corresponding with the chosen attribute to change. They then use an attribute vector orthogonal to the classifier's boundary to traverse different levels of the chosen attribute. In order to analyze the classification of their generated counterfactuals, they define a metric of bias called *image counterfactual sensitivity analysis*, which measures the sensitivity of one attribute classification to the changes of another in the image.

Zhou et al. 2021 explore various machine learning explanation methods including metrics on their interpretability and fidelity. Wang et al. 2021, Goyal et al. 2019, and Mothilal et al. 2020 all explore various counterfactual explanations both image and non-image.

Our project adopts a similar approach to find bias as Dash et al. 2022 and Denton et al. 2020, while using a different method to create our image counterfactuals that can be applied easily to other classification models. Our project also focuses on a more holistic approach to measuring and quantifying bias, looking at several different perspectives of the definition of fairness and bias. Unlike Dash et al., we measure the bias of our counterfactuals on many attributes, across multiple definitions.

3. Methods

The first step of the process in this project was to create image counterfactuals of faces by changing a single attribute in an image. In order to do this, we used FaderNetworks. FaderNetworks uses an encoder-decoder architecture, where the encoder takes in an input image and its associated attributes, and maps them to a latent representation. The decoder is then trained to reconstruct the input image given this latent representation and attributes. At inference time, a test image is encoded in the latent space, and the user can choose the attribute values to be fed to the decoder, which will then generate a reconstruction of the image based on those attributes. Even though the attributes are originally binary, FaderNetworks considers the attribute

a continuous variable during inference so that a sliding scale can be used for the final image. FaderNetworks is trained on the CelebA dataset which includes 202,599 faces and 40 attribute labels. Using FaderNetworks, we created 10 interpolations using 10 faces over three chosen attributes: “Young”, “Male”, and “Eyeglasses”. After creating interpolations with FaderNetworks, we classified them. FaderNetworks includes a pre-trained classification model trained on celebA that we used to classify the counterfactual images. We ran the outputs through a sigmoid function to ensure all outputs were between 0 and 1.

In order to determine how successful FaderNetworks’s encoder-decoder reconstruction process is, we measured the error between the original images and their closest corresponding counterpart in the interpolations. Assuming the reconstruction process does generate too much noise in the image, the two faces should be nearly identical and therefore have the same classifications for all other attributes. By measuring the difference in these attributes we can estimate an error for the accuracy of FaderNetworks. We used the mean squared error to congregate the differences and calculate error for each of the attributes with we chose to create interpolations, the results of which can be seen in table 1. The creators of FaderNetworks Lample et al. also found that their FaderNetworks-generated images had a fairly high human-rated naturalness score of 88.4%, 75.2% and 78.8% for three of the reconstructed images (in comparison to a 90% rating of naturalness for the real faces). They also found that the generated attribute swaps had very high classification scores with 76.6% accuracy for mouth open and 97.1% accuracy for smile. Another note on the accuracy of FaderNetworks is that if the interpolations themselves introduce bias into the generated images, then it still reflects bias that was already present in the models learning from this data.

After determining whether FaderNetworks was a valid method to adjust a single attribute and create counterfactual images, we then were able to use the outputs to measure the bias of the classification model. Bias in machine learning models can arise from several different sources (Hellstrom et al., 2020). Bias can come from the learning of the model if hyper-parameters, architecture, and model design are not chosen well. Historical bias is caused by bias present and measured in the world as it is. Bias often comes from data generation through problems in sampling, annotating, and inheriting bias from data generated from another biased model. This experiment will be using a holistic definition of bias without diving into the original source of the bias.

Not only are there different sources of bias, but there are also different interpretations of bias depending on how fairness is defined. For this experiment we measured four different interpretations of bias based on different definitions of fairness. The first definition of bias we used was co-occurrence bias (Hellstrom et al., 2020). Because many machine learning techniques rely on identifying correlations in data, these biases can be propagated in the models or classifiers that are learned from the data. Co-occurrence bias is a method of detecting and measuring this bias related to correlation. We measure co-occurrence bias as $b(o, g)$ below. With an attribute classification being fair when $b(o, g)$= 0.

$$b(o, g) = \frac{c(o, g)}{\sum_{g' \epsilon G} c(o, g')'}$$

The next three definitions of fairness we use are demographic parity, equal opportunity and equalized odds. We used those three definitions of fairness in terms of constraint functions that serve as linear constraints on the model *h* to define bias. Defining the classification model as $h: X \rightarrow [0,1]$, we define a fair model when

$$E_{x \sim P}[\langle h(x), c(x) \rangle] = 0$$

where $\langle h(x), c(x) \rangle = h(x), c(x, y)$ and $h(x)$ denotes the probability of sampling y from a Bernoulli random variable with p = h(x). The model h is then unbiased with respect to the constraint when $E_{x \sim P}[\langle h(x), c(x) \rangle] = 0$ and the degree of bias is given by $E_{x \sim P}[\langle h(x), c(x) \rangle]$. Below we list the different constraint functions for each definition of fairness. We define g(x) as an indicator function $g(x) = 1[x \in G]$. We use $Z_g := E_{x \sim P}[g(x)]$ to denote the probability of a sample drawn from P to be in G. We use $P_x = E_{x \sim P}[ytrue(x)]$ to denote the proportion of X which is positively labeled and $P_g := E_{x \sim P}[g(x) * ytrue(x)]$ to denote the proportion of X which is positively labeled and in G. In order to define $ytrue(x)$ which is defined as a theoretical unbiased ground truth label function $X \rightarrow [0,1]$, we considered the top 25% of classification values in the respective attribute being measured as an estimate of real why true so that when a classification fell within that measure $ytrue(x) = 1$, with the bottom 75% representing $ytrue(x) = 0$. In order to determine whether an image was in a protected class i.e. $g(x) = 1$, we defined a boundary as the average of all classifications for that protected attribute class. Images that had classification values that exceeded that boundary were classified as in the protected class.

Demographic parity (Dwork et al., 2012): Demographic parity is most similar to co-occurrence bias. A fair classifier h should make positive predictions independently of the protected attribute. The constraint function may be expressed as

$$c(x,0) = 0, \qquad c(x,1) = \frac{g(x)}{Z_G} - 1$$

Equalized odds (Hardt et al., 2016): A classifier h is fair if h and the protected attribute are independent conditional on the target variable Y. h can depend on the protected attribute , but only through the target variable. In addition to the constraint associated with equal opportunity, this notion applies an additional constraint with

$$c(x,0) = 0, \qquad c(x,1) = \frac{g(x)\big(1 - ytrue(x)\big)}{Z_G - P_G} - \frac{1 - ytrue(x)}{1 - P_x}$$

Equal opportunity (Hardt et al., 2016): Equalized opportunity is a relaxed version of equalized odds. A fair classifier h requires non-discrimination only within the advantaged outcome group, advantaged defined as the more "desirable" outcome group, for example, being classified as "attractive" being more desirable than not being classified as "attractive". The constraint may be expressed as

$$c(x,0) = 0, \qquad c(x,1) = \frac{g(x)ytrue(x)}{P_G} - \frac{ytrue(x)}{P_x}$$

For the purposes of this project we simply define the positive classification as the advantage outcome, whether or not it is seen as more "desirable".

4. Results:

Error:

Table 1: FaderNetworks MSE

Attribute	Young	Male	Eyeglasses
Error	**1.29%**	**3.44%**	**0.95%**

FaderNetworks Interpolations

Young Interpolations

Figure 1: 10 different faces with 10 counterfactual interpolations over the young attribute, the first image is the original, the second is the reconstructed image, the rest are the interpolations.

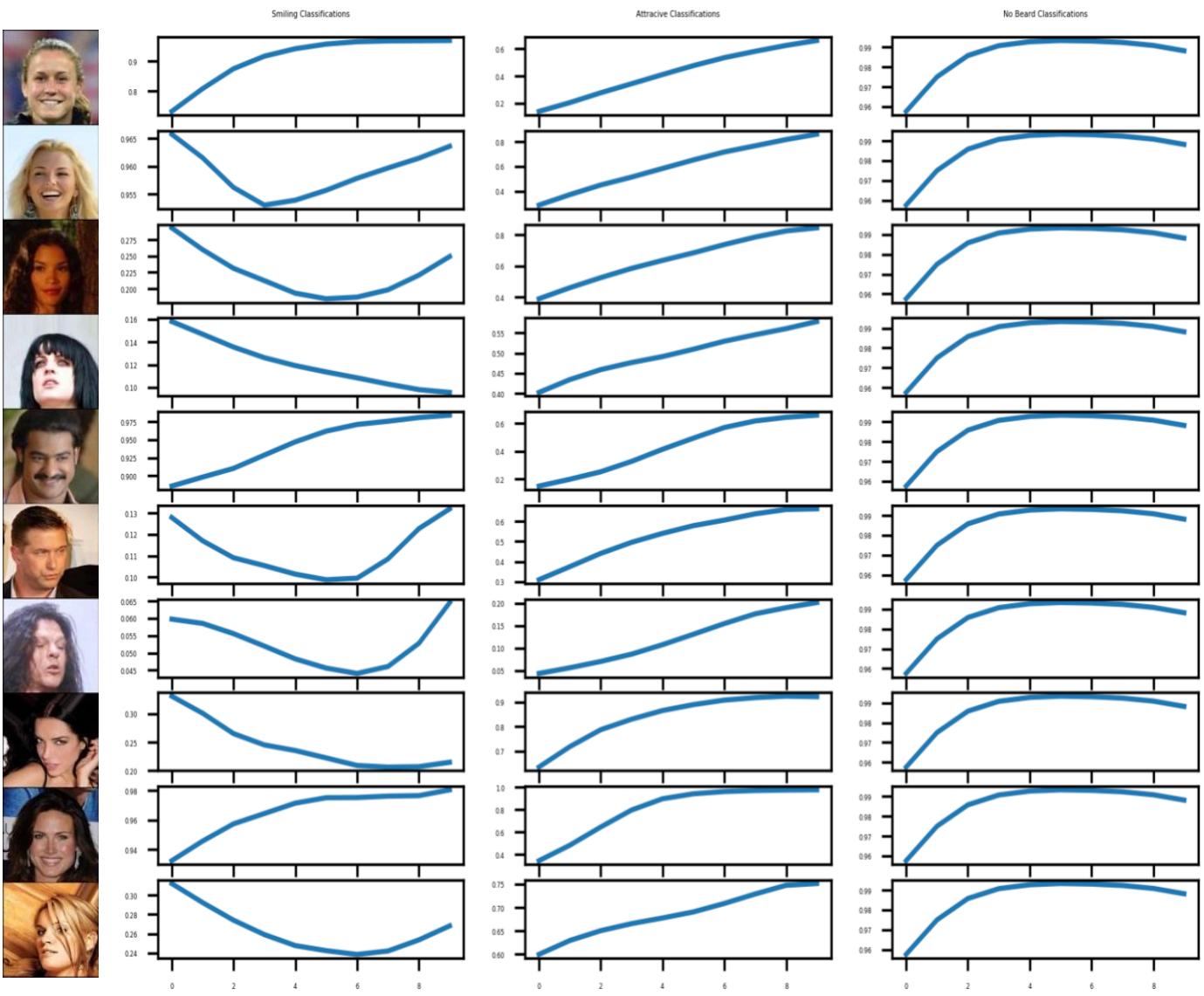

Figure 2: Young interpolation classifications for each face. Classifications are for the smiling, attractive, and no beard attributes over all 10 young counterfactual images.

Male Interpolations

Figure 3: 10 different faces with 10 counterfactual interpolations over the male attribute, the first image is the original, the second is the reconstructed image, the rest are the interpolations.

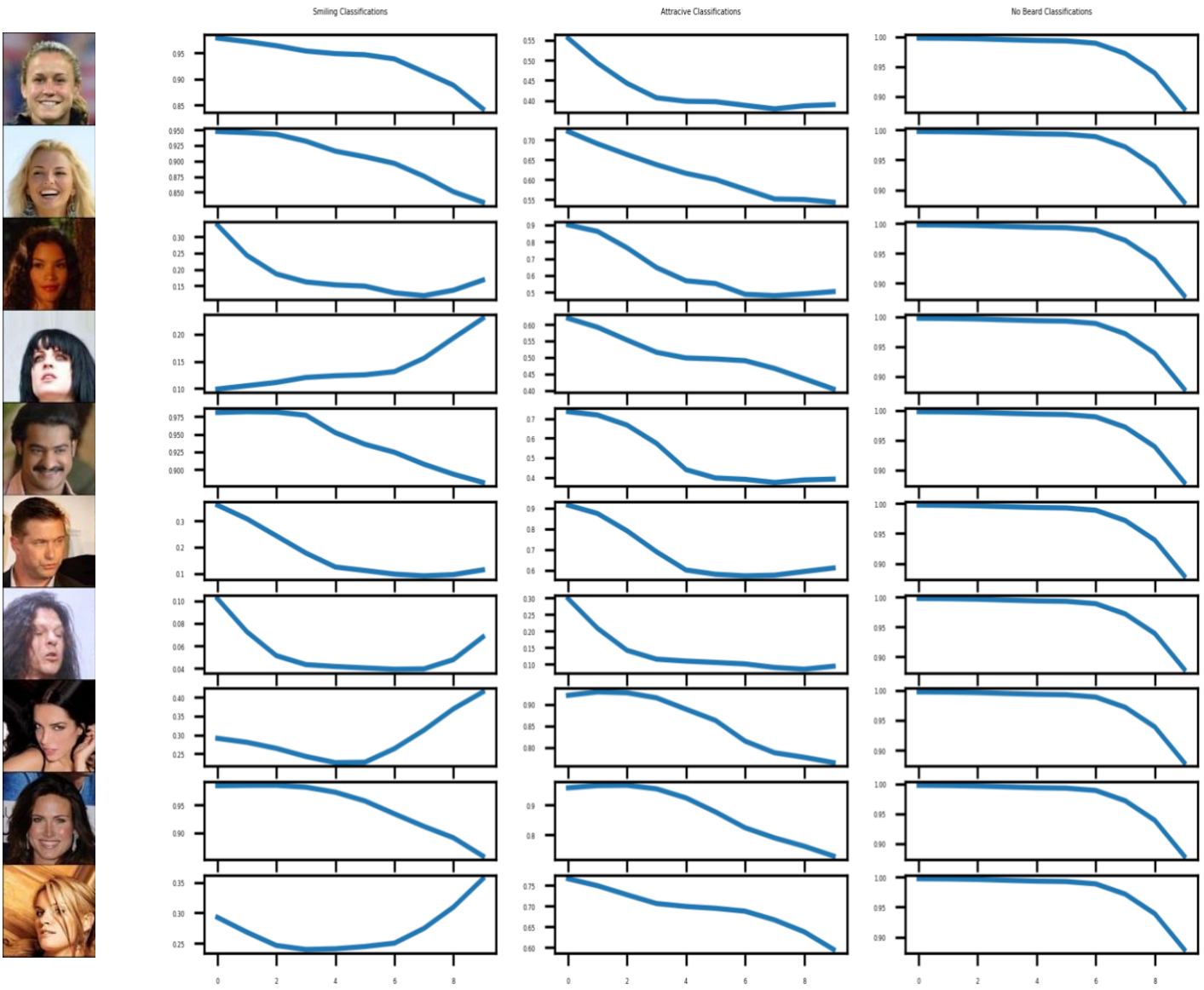

Figure 4: Male interpolation classifications for each face. Classifications are for the smiling, attractive, and no beard attributes over all 10 young counterfactual images.

Eyeglasses Interpolations

Figure 5: 10 different faces with 10 counterfactual interpolations over the eyeglasses attribute, the first image is the original, the second is the reconstructed image, the rest are the interpolations.

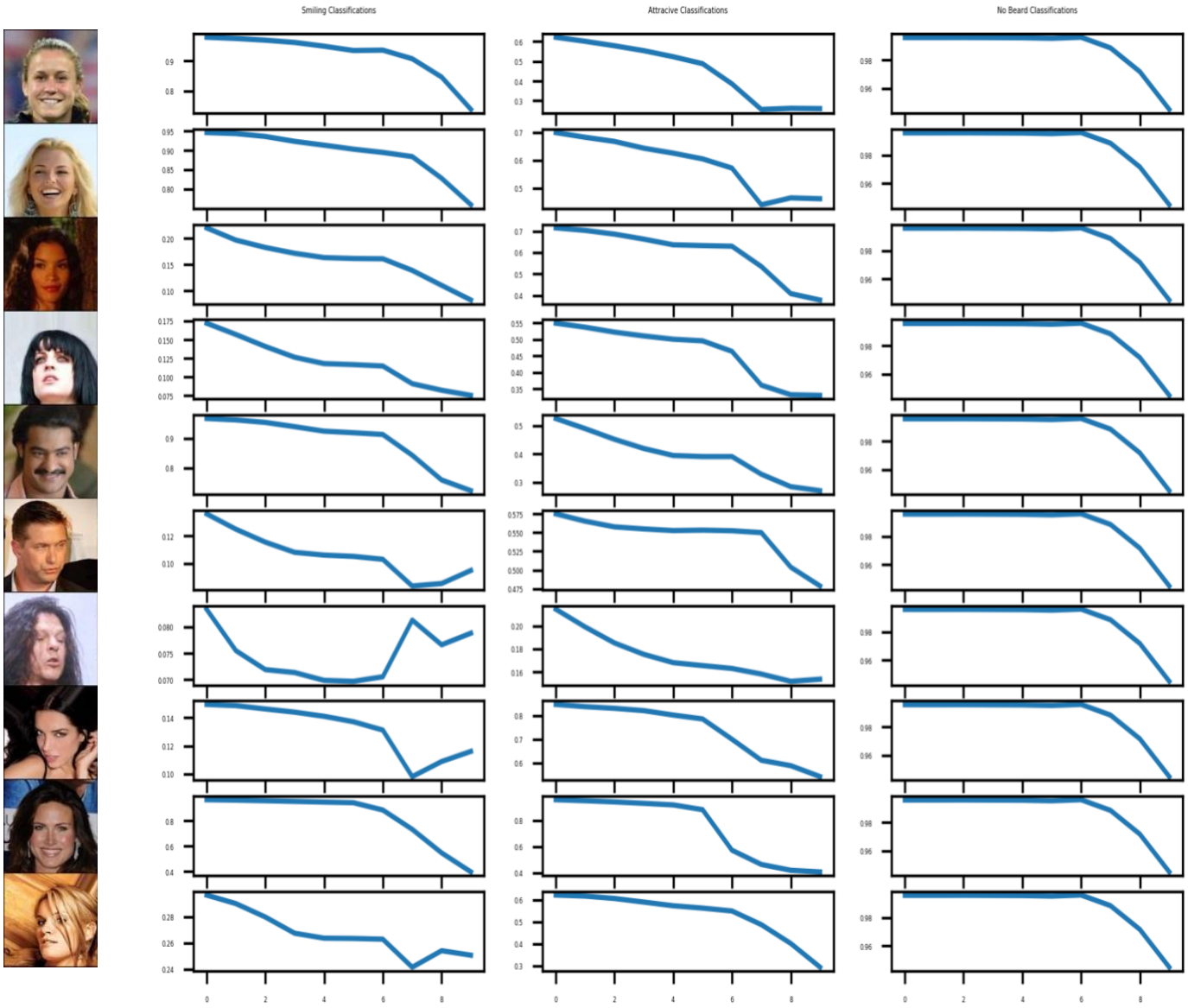

Figure 6: Eyeglasses interpolation classifications for each face. Classifications are for the smiling, attractive, and no beard attributes over all 10 young counterfactual images.

Bias Calculations

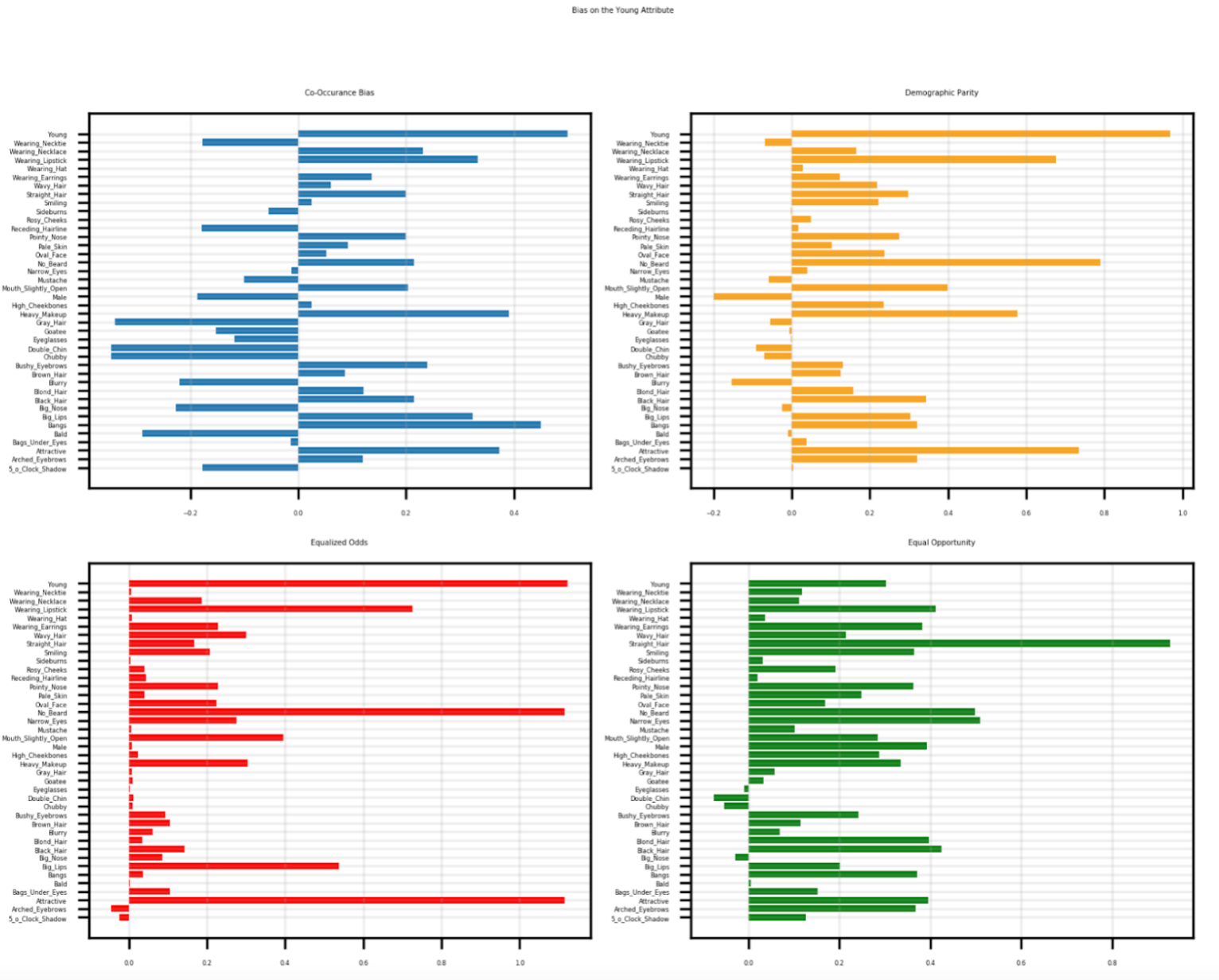

Figure 4: Various levels of each type of bias over all attributes with respect to the young counterfactual images.

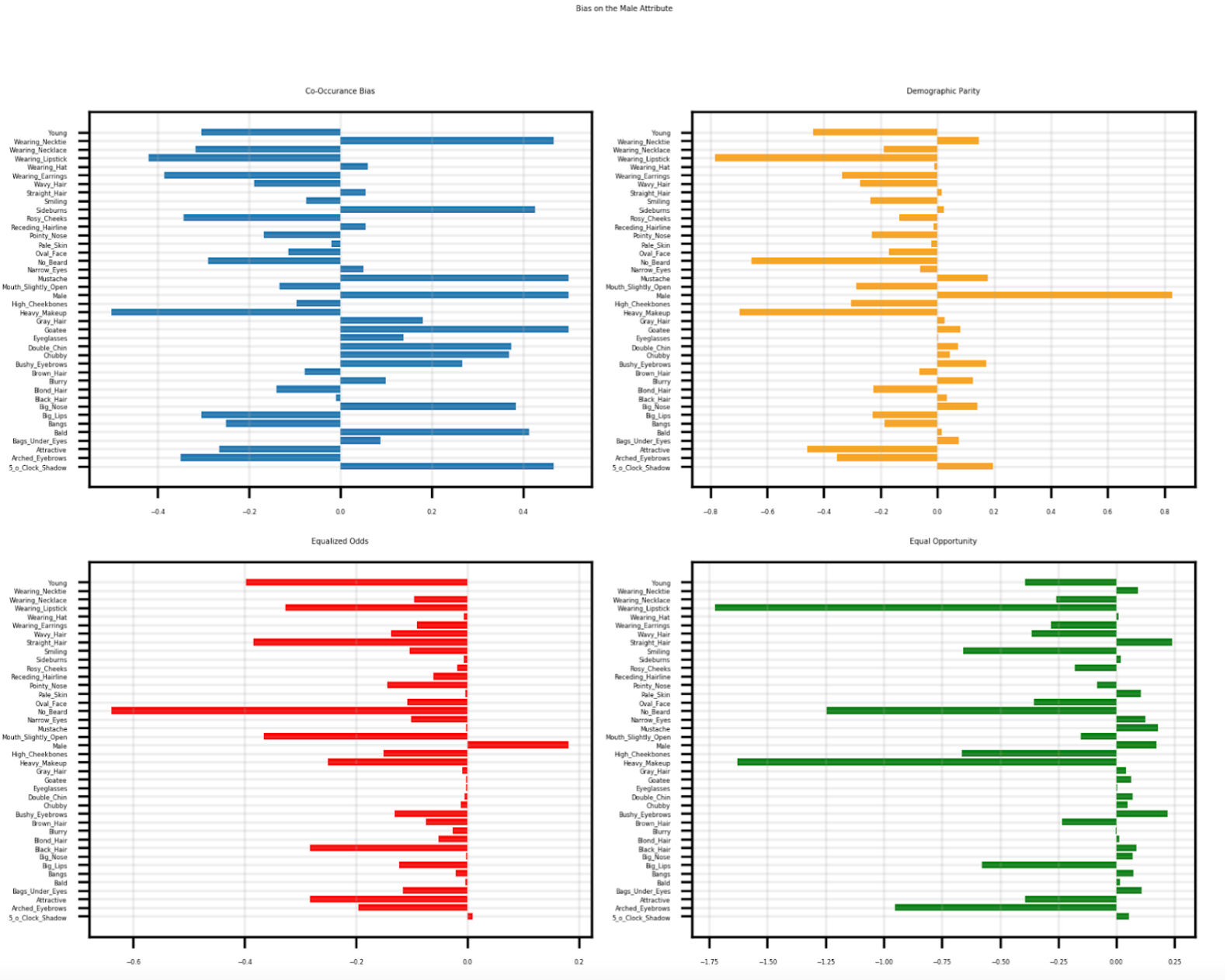

Figure 5: Various levels of each type of bias over all attributes with respect to the male counterfactual images.

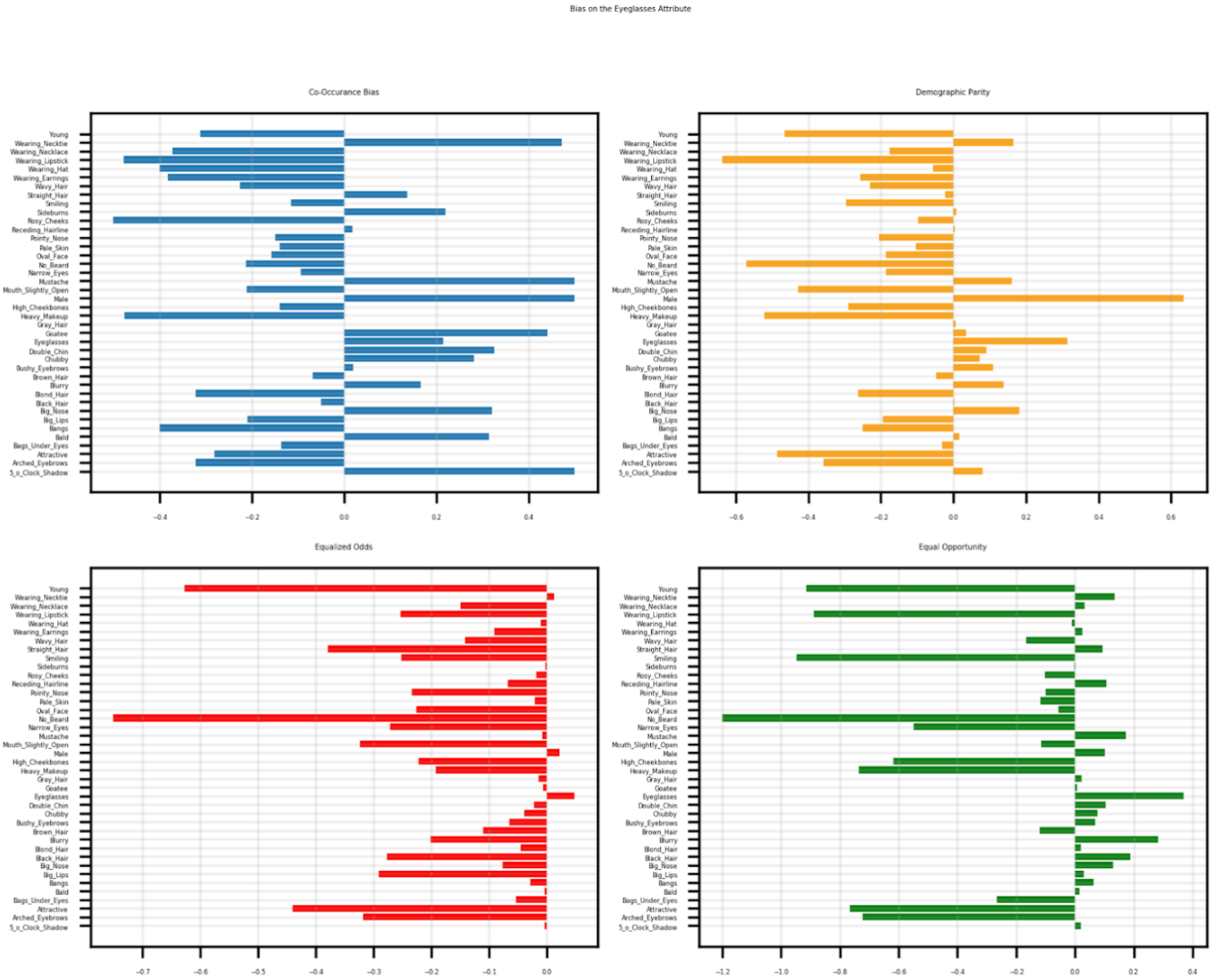

Figure 6: Various levels of each type of bias over all attributes with respect to the eyeglasses counterfactual images.

5. Discussion

Error

As seen in table 1, the error we measured in counterfactual generation was small, with all 3 attribute interpolations having error < 3.5%. This confirms that in generating the counterfactual images, significant noise is not being generated, and the changes in the image are isolated to the attribute we wish to change. Looking at the images themselves, it's clear that there is variation in the success and consistency of the counterfactuals across interpolations. The eyeglasses interpolations, for example, do not form very realistic mid-level images. This is due to the fact that wearing glasses is a binary feature, so no realistic image would exist of a face "half" wearing glasses. The other two attributes we apply FaderNetworks to, young and male, exist in the world on more of a spectrum and therefore those attributes have more realistic intermediary images.

Bias

Considering our classification model was trained on the CelebA dataset which is known to lack diversity and contain some subjective attributes such as "attractive" (Wang et al., 2021, Rajabi et al. 2022), we expected to find some substantial and interesting bias in our analysis, and this was certainly the case.

Co-occurrence bias

Co-occurrence bias is defined as measuring possible correlations between attributes in the training data that have been propagated through the model in training. Co-occurrence bias seems to have quite a few of the intuitive biases we might expect, such as a negative bias for gray hair and receding hairline with respect to the young attribute, and negative bias for heavy

makeup with respect to the male attribute. Based on the definition of co-occurrence bias, these are some biases we might expect as there are likely correlations between gray hair and a receding hairline with being old in the data's labels. Some interesting biases include a very high positive bias for bangs with respect to the young attribute, and a fairly high negative bias for young with respect to the male attribute. Another interesting observation is that for the eyeglasses interpolations, the male attribute has a very strong positive bias. It seems that because of this the eyeglasses counterfactuals also have many similar positive and negative biases that the male counterfactuals have.

One of the more interesting attributes to analyze the bias of is smiling, because theoretically in a fair classifier, smiling should remain the same as the age, gender, and eyeglasses attributes change. Smiling appears to have a slight positive bias with respect to the young attribute and a negative bias with respect to the male and eyeglasses attributes. According to the definition of co-occurrence bias, this means that there is some sort of correlation between young and smiling along with male and eyeglasses with not smiling being propagated through the model, and that with this particular model, if a face becomes older or more male presenting they will be classified as smiling less.

Demographic Parity

Demographic parity and the next two definitions of bias are defined with the intentions of mitigating said bias to create a fair classifier. Demographic parity at first appears to be the most similar to co-occurrence bias. A fair classifier with demographic parity is defined as having the same proportion of positive classifications on a face with the protected attribute as would any face in the data set. In other words, the decision of the classifier should be independent of the protected attribute. Young has some similar high biases that it had in co-occurrence such as no

beard, attractive, and wearing lipstick, but it does not have the same strong negative biases. In fact, the young attribute has very few negative biases at all. In contrast, the male and eyeglasses attributes have mainly strong negative biases. Similarly to co-occurrence bias, male and eyeglasses have some very similar biases. This is particularly interesting for the eyeglasses attribute as for the young and male attributes, the highest positive bias is the attribute being changed through counterfactuals, while for the eyeglasses attribute, the highest positive bias is toward the male attribute.

Looking at the smiling attribute, there is a slight positive bias toward the young counterfactuals, while there is almost no bias measured for the male counterfactuals. There is however a significant negative bias for smiling toward wearing eyeglasses. This might be a bias to consider mitigating as the attributes young and wearing eyeglasses should not have an effect on the classification of smiling.

Equalized odds

Equalized odds allows a fair model to depend on the sensitive attribute, but only through the target variable. Similarly to demographic parity but to more of an extreme, the equalized odds bias for the male and eyeglasses counterfactuals are almost all negative while young are almost all positive. We assume that the reason for this is that with equalized odds and equal opportunity, we define a positive classification of all of our attributes as being the "desired" classification in our analysis of bias. This definition attempts to measure what classifications should be increased, or decreased. In this case, the attributes male and eyeglasses seem to decrease the odds of classification in most of these attributes while the young attribute seems to increase the odds. In order to mitigate these odds, we would have to adjust the model to offset their effects.

The male counterfactuals have some very strong negative biases that one might expect such as no beard, but interestingly also have a very strong negative bias toward mouth open and attractive attributes. The eyeglasses counterfactuals also have a very strong negative bias with respect to no beard and attractive attributes. The attribute no beard seemed to have a strong sensitivity in this measure of bias, having a very strong bias, both negative and positive in all three examples of counterfactuals. According to the definition of equalized odds this sensitivity is due to the fact that the model depends greatly on the sensitive attribute through the no beard attribute. The no beard attribute having a negative bias toward the male interpolations seems intuitively to make sense, but is more interesting to observe for the eyeglasses interpolations.

Equal Opportunity

Equal opportunity is a relaxed version of equal odds that requires non-discrimination only within the "advantaged" classification. Similarly to equalized odds, male and eyeglasses have mostly negative biases associated with them, while young is associated with mostly positive bias. Interestingly, there is a very high bias toward straight hair for the young attribute. There is a very low bias toward heavy makeup and wearing lipstick for the male attribute. There is also a very low bias for young, no beard, and wearing lipstick with respect to the eyeglasses attribute. Looking at the smiling attribute, the results for bias are very similar to equalized odds in this measure.

Due to the similarities of the male and wearing eyeglasses bias graphs for demographic parity, equalized odds, and equal opportunity, we might assume that eyeglasses would have a strong positive bias with respect to the male attribute. Interestingly, for the biases of the male counterfactuals, eyeglasses seem to have negligible bias.

Ultimately in using FaderNetworks to create counterfactual images of faces and using a diverse set of definitions of bias, we were able to calculate and analyze several different kinds of biases in this model. This technique successfully found and measured bias and could be applied to other image classification models to find and analyze their biases.

6. Future Work:

Ultimately with this project, we found a method to find and quantify the types of bias in an image classification model. The next logical step would be to find a method to mitigate the bias that you would wish to not have in a model. In order to mitigate bias you would first need to find a method to determine what bias you want to keep and what bias you actually would like to be mitigated.

Some methods of mitigating bias ignore the actual source of bias, but another direction that this project could take would be to explore methods of discovering the source of the bias, whether it be the data or the model.

Another direction this project could take is to use the counterfactuals on different and more impactful classification models. With more resources and time it would be interesting to measure possible bias within those models.

Works Cited

Dash, Saloni, et al. *Evaluating and Mitigating Bias in Image Classifiers: A Causal Perspective Using Counterfactuals*. 2022.

Denton, Emily, et al. *Image Counterfactual Sensitivity Analysis for DetectingUnintended Bias*. 2020.

"Enhanced Video Analytics (EVA) - Biometric Solutions - NEC New Zealand." *NEC*, 2020, www.nec.co.nz/expertise/biometrics/enhanced-video-analytics/. Accessed 22 Dec. 2022.

Garbie, Clare, and Jonathan Frankle. *Facial-Recognition Software Might Have a Racial Bias Problem*. 2016.

Goyal, Yash, et al. *Counterfactual Visual Explanations*. 2019.

Group, Vantage Technology Consulting. "What Are the Key Use Cases for Facial Recognition in Healthcare?" *Vantage Technology Consulting Group*, 15 Aug. 2019, www.vantagetcg.com/what-are-the-key-use-cases-for-facial-recognition-in-healthcare. Accessed 22 Dec. 2022.

Guillame, Lample et al. *Fader Networks: Manipulating Images by Sliding Attributes*. 2018.

Halamka, John, et al. "Addressing Racial Disparities in Surgical Care with Machine Learning." *Npj Digital Medicine*, vol. 5, no. 1, 30 Sept. 2022, 10.1038/s41746-022-00695-6. Accessed 9 Oct. 2022.

Hellström, Thomas, et al. *Bias in Machine Learning - What Is It Good For?* 2020.

Moritz, Hardt, et al. *Equality of Opportunity in Supervised Learning*. 2016.

Mothilal, Ramaravind K., et al. “Explaining Machine Learning Classifiers through Diverse Counterfactual Explanations.” *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, 22 Jan. 2020, 10.1145/3351095.3372850.

Office, U. S. Government Accountability. “Facial Recognition Technology: Current and Planned Uses by Federal Agencies.” *Www.gao.gov*, 24 Aug. 2021, www.gao.gov/products/gao-21-526.

Rajabi, Amirarsalan et al. *Through a Fair Looking-Glass: Mitigating Bias in Image Datasets*. 2022.

Verma, Sahil, et al. *Counterfactual Explanations for Machine Learning: A Review*. 2022.

Wang, Cong, et al. “Counterfactual Explanations in Explainable AI: A Tutorial.” *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 14 Aug. 2021, 10.1145/3447548.3470797.

Wang, Xiaoyang, et al. *Robust and Personalized Federated Learning with Spurious Features: An Adversarial Approach*. 2021.

Zhou, Jianlong, et al. “Evaluating the Quality of Machine Learning Explanations: A Survey on Methods and Metrics.” *Electronics*, vol. 10, no. 5, 4 Mar. 2021, p. 593, res.mdpi.com/electronics/electronics-10-00593/article_deploy/electronics-10-00593-v3.pdf, 10.3390/electronics10050593. Accessed 18 Apr. 2021.

Appendix

The FaderNetworks implementation we used can be found at https://github.com/facebookresearch/FaderNetworks, although code must be modified for current PyTorch versions.

Python Implementation

```
# Imports

import numpy as np
import torch
import os
import matplotlib.pyplot as plt

# import FaderNetworks Directory
import sys
sys.path.append('../FaderNetworks')

from src.logger import create_logger
from src.loader import load_images, DataSampler
from src.utils import bool_flag
from src.loader import AVAILABLE_ATTR

# Classification

def sig(x):
    return 1/(1 + np.exp(-x))

# Use classifier to classify counterfactual interpolations
def classifications(model, interpolations):
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    mod = torch.load(model).eval()
    interps = torch.load(interpolations).to(device)
    outputs = []
    for x in interps:
        outputs.append(mod(x))

    # Take the second value in each pair
    new_outputs = []
    for tensor1 in outputs:
        tens1 = []
        for tensor2 in tensor1:
            tens2 = []
            for x in range(0,80,2):
                tens2.append(sig(tensor2[x+1].cpu().detach().numpy()))
            tens1.append(tens2)
        new_outputs.append(tens1)

    return new_outputs
```

```
# Get the classification outputs for the interpolations for one single
attribute without including the first 2 images
def isolate_attribute(outputs, attribute):
    attribute_classifications = []

    for i in outputs:
        face = []
        for x in i:
            face.append(x[AVAILABLE_ATTR.index(attribute)]) #not including
the first 2 images
        attribute_classifications.append(face[2:])

    return attribute_classifications

# Isolate a single attribute along with the original images
def isolate_attribute_with_orig(outputs, attribute):
    attribute_classifications = []

    for i in outputs:
        face = []
        for x in i:
            face.append(x[AVAILABLE_ATTR.index(attribute)]) #including the
first 2 images
        attribute_classifications.append(face)

    return attribute_classifications



# MSE

# Calculate MSE error in FaderNetworks
# Compare the classifciation values of the first face to the interpolated
face with the most similar value

def error(outputs, attr):
    attribute_classifications = []

    # Find the faces most similar to the original in the grid
    for i in outputs:
        face = []
        for x in i:
            face.append(x[AVAILABLE_ATTR.index(attr)])
        attribute_classifications.append(face)

    face_index_arr = []
    for face in attribute_classifications:
        orig = face[0]
        diff = 999
        curr = 2
        for i in range(2,12):
            if(abs(face[i]- orig) < diff):
                diff = face[i]- orig
                curr = i
        face_index_arr.append(curr)
```

```
    # Sum all the error
    sum = 0
    total = 0
    for attribute in AVAILABLE_ATTR:
        attribute_classifications = isolate_attribute_with_orig(outputs,
attribute)
        ind = 0
        for face in attribute_classifications:
            sum += (face[0] - face[face_index_arr[ind]])**2
            ind+=1
            total+=1

    return (sum/total) * 100


# Bias Definitions:

def get_attr_boundary(outputs, prot_attr):

    attr_classifications = isolate_attribute(outputs, prot_attr)
    boundary = np.average(attr_classifications)

    attr_boundary = []
    for face in attr_classifications:
        for i in range(0,10):
            if face[i] >= boundary:
                attr_boundary.append(i)
                break
            if i == 9:
                attr_boundary.append(8)
    return attr_boundary


def co_occurrence_bias(outputs, attribute):
    biases = []
    prot_boundary = get_attr_boundary(outputs, attribute)

    for attribute in AVAILABLE_ATTR:
        attribute_classifications = isolate_attribute(outputs, attribute)
        boundary = np.average(attribute_classifications)


        g = 0
        g_prime = 0
        ind = 0
        for face in attribute_classifications:
            for i in range(0,prot_boundary[ind]):
                if face[i] > boundary:
                    g_prime+=1
            for i in range(prot_boundary[ind],10):
                if face[i] > boundary:
                    g+=1
            ind+=1
        bias_score = (g / (g + g_prime)) - (1/2)
```

```
        biases.append(bias_score)
    return biases


def demographic_parity(outputs, prot_attr):
    biases = []
    g_boundary = get_attr_boundary(outputs, prot_attr)

    for attribute in AVAILABLE_ATTR:
        attribute_classifications = isolate_attribute(outputs, attribute)

        exp_sum = 0
        zg = 0.5
        ind = 0
        for face in attribute_classifications:
            for i in range(0,g_boundary[ind]): #g(x) = 0
                exp_sum += face[i]*(-1)
            for i in range(g_boundary[ind],10): #g(x) = 1
                exp_sum += face[i]*(1/zg - 1)
            ind+=1

        biases.append(exp_sum/40)

    return biases



def equal_opportunity(outputs, prot_attr):
    biases = []
    g_boundary = get_attr_boundary(outputs, prot_attr)

    for attribute in AVAILABLE_ATTR:
        attribute_classifications = isolate_attribute(outputs, attribute)
        true_boundary = np.percentile(attribute_classifications, 75)


        exp_sum = 0
        ytrue = 1
        ind = 0
        for face in attribute_classifications:
            px = 0
            pg = 0
            for i in range(0,10):
                if face[i] > true_boundary:
                    px+=1
                    if i >= g_boundary[ind]:
                        pg+=1

            px = px/10
            pg = pg/10
            ind = 0
            for i in range(0,g_boundary[ind]):#g(x) = 0
                if(face[i] > true_boundary):
                    exp_sum += face[i]*(-(ytrue)/px)
            for i in range(g_boundary[ind],10):#g(x) = 1
                if(face[i] > true_boundary and pg != 0):
```

```
                    exp_sum += face[i]*(ytrue/pg - ytrue/px)
                #exp_sum += face[i]*(ytrue/0.5 - ytrue/px)
            ind+=1

        biases.append(exp_sum/40)

    return biases


def equalized_odds(outputs, prot_attr):
    biases = []
    g_boundary = get_attr_boundary(outputs, prot_attr)

    for attribute in AVAILABLE_ATTR:
        attribute_classifications = isolate_attribute(outputs, attribute)
        true_boundary = np.percentile(attribute_classifications, 75)


        exp_sum = 0
        zg = 0.5
        ind = 0
        for face in attribute_classifications:
            px = 0
            pg = 0
            for i in range(0,10):
                if face[i] > true_boundary:
                    px+=1
                    if i > 4:
                        pg+=1
            px = px/10
            pg = pg/10
            ind = 0
            for i in range(0,g_boundary[ind]): #g(x) = 0
                if(face[i] < true_boundary):
                    exp_sum += face[i]*(-(1)/(1-px))
            for i in range(g_boundary[ind],10): #g(x) = 1
                if(face[i] < true_boundary and (zg - pg) != 0):
                    exp_sum += face[i]*((1)/(zg - pg) - (1)/(1-px))
            ind+=1

        biases.append(exp_sum/40)

    return biases

# Plotting

def plot_attribute_grid(outputs):
    fig, axs = plt.subplots(10, 3)

    #2, 24, 31 attractive, no_beard, and smiling

    smiling_classifications = isolate_attribute(outputs, "Smiling")
    attractive_classifications = isolate_attribute(outputs, "Attractive")
    no_beard_classifications = isolate_attribute(outputs, "No_Beard")

    axs[0,0].plot(smiling_classifications[0])
```

```
    axs[0, 0].set_title('Smiling Classifications')
    axs[1,0].plot(smiling_classifications[1])
    axs[2,0].plot(smiling_classifications[2])
    axs[3,0].plot(smiling_classifications[3])
    axs[4,0].plot(smiling_classifications[4])
    axs[5,0].plot(smiling_classifications[5])
    axs[6,0].plot(smiling_classifications[6])
    axs[7,0].plot(smiling_classifications[7])
    axs[8,0].plot(smiling_classifications[8])
    axs[9,0].plot(smiling_classifications[9])

    axs[0,1].plot(attractive_classifications[0])
    axs[0, 1].set_title('Attracive Classifications')
    axs[1,1].plot(attractive_classifications[1])
    axs[2,1].plot(attractive_classifications[2])
    axs[3,1].plot(attractive_classifications[3])
    axs[4,1].plot(attractive_classifications[4])
    axs[5,1].plot(attractive_classifications[5])
    axs[6,1].plot(attractive_classifications[6])
    axs[7,1].plot(attractive_classifications[7])
    axs[8,1].plot(attractive_classifications[8])
    axs[9,1].plot(attractive_classifications[9])

    axs[0,2].plot(no_beard_classifications[0])
    axs[0, 2].set_title('No Beard Classifications')
    axs[1,2].plot(no_beard_classifications[0])
    axs[2,2].plot(no_beard_classifications[0])
    axs[3,2].plot(no_beard_classifications[0])
    axs[4,2].plot(no_beard_classifications[0])
    axs[5,2].plot(no_beard_classifications[0])
    axs[6,2].plot(no_beard_classifications[0])
    axs[7,2].plot(no_beard_classifications[0])
    axs[8,2].plot(no_beard_classifications[0])
    axs[9,2].plot(no_beard_classifications[0])



def plot_attribute_bias(biases, title, color):
    x = list(range(40))
    plt.barh(x,biases, tick_label=AVAILABLE_ATTR, color = color)
    plt.title(title)
    plt.grid()

def plot_multiple_bias(bias1, bias2, bias3, bias4):

    width = 0.3
    x = np.arange(40)

    xvals = bias1
    bar1 = plt.barh(x, xvals, width, color = 'r')

    yvals = bias2
    bar2 = plt.barh(x+width, yvals, width, color='g')

    zvals = bias3
    bar3 = plt.barh(x+width*2, zvals, width, color = 'b')
```

```
    avals = bias4
    bar4 = plt.barh(x+width*3, avals, width, color = 'y')

    #plt.xlabel("Dates")
    plt.ylabel('Scores')
    plt.title("Young Interpolations Bias")

    plt.yticks(x+width, AVAILABLE_ATTR)
    plt.legend( (bar1, bar2, bar3,bar4), ('co-occurence', 'Demographic
Parity', 'Equal Opportunity', 'Equalized Odds') )
    plt.show()

def plot_grid(bar1, bar2, bar3, bar4, title):
    x = np.arange(40)
    fig, axs = plt.subplots(2, 2)
    fig.suptitle(title)

    axs[0, 0].barh(x, bar1, tick_label = AVAILABLE_ATTR)
    axs[0, 0].set_title('Co-Occurance Bias')
    axs[0, 0].grid(which='minor', alpha=0.1)
    axs[0, 0].grid(which='major', alpha=0.2)
    axs[0, 1].barh(x, bar2, color = 'orange', tick_label = AVAILABLE_ATTR)
    axs[0, 1].set_title('Demographic Parity')
    axs[0, 1].grid(which='minor', alpha=0.1)
    axs[0, 1].grid(which='major', alpha=0.2)
    axs[1, 1].barh(x, bar3, color = 'green', tick_label = AVAILABLE_ATTR)
    axs[1, 1].set_title('Equal Opportunity')
    axs[1, 1].grid(which='minor', alpha=0.1)
    axs[1, 1].grid(which='major', alpha=0.2)
    axs[1, 0].barh(x, bar4, color = 'red', tick_label = AVAILABLE_ATTR)
    axs[1, 0].set_title('Equalized Odds')
    axs[1, 0].grid(which='minor', alpha=0.1)
    axs[1, 0].grid(which='major', alpha=0.2)



#Young Plots

young_outputs = classifications('classifier256.pth',
'young_interpolations.pth')

co_bias = co_occurrence_bias(young_outputs, "Young")
dem_par = demographic_parity(young_outputs, "Young")
eq_opp = equal_opportunity(young_outputs, "Young")
eq_odds = equalized_odds(young_outputs, "Young")

plot_grid(co_bias, dem_par, eq_opp, eq_odds, "Bias on the Young Attribute")
plot_attribute_grid(young_outputs)

#Male Plots

male_outputs = classifications('classifier256.pth',
'male_interpolations.pth')

co_bias = co_occurrence_bias(male_outputs, "Male")
dem_par = demographic_parity(male_outputs, "Male")
```

```
eq_opp = equal_opportunity(male_outputs, "Male")
eq_odds = equalized_odds(male_outputs, "Male")

plot_grid(co_bias, dem_par, eq_opp, eq_odds, "Bias on the Male Attribute")
plot_attribute_grid(male_outputs)
```

#Eyeglasses Plots

```
eyeglasses_outputs = classifications('classifier256.pth',
'eyeglasses_interpolations.pth')

co_bias = co_occurrence_bias(eyeglasses_outputs, "Male")
dem_par = demographic_parity(eyeglasses_outputs, "Male")
eq_opp = equal_opportunity(eyeglasses_outputs, "Male")
eq_odds = equalized_odds(eyeglasses_outputs, "Male")

plot_grid(co_bias, dem_par, eq_opp, eq_odds, "Bias on the Eyeglasses
Attribute")
plot_attribute_grid(eyeglasses_outputs)
```