IJHSR
ijhighschoolresearch.org
RESEARCH ARTICLE

Reducing Racial and Gender Bias in Machine Learning and Natural Language Processing Tasks Using a GAN Approach

Isabella S. Mandis

The Spence School, 22 East 91st Street, New York City, NY 10128, USA; IMandis22@spenceschool.org

ABSTRACT: While organizations such as Amazon have used machine learning-based algorithms for their hiring processes, diverse employees are not equitably hired due to biased datasets. Current approaches to debias machine learning algorithms are expensive and difficult to implement. This research uses a Generative Adversarial Network (GAN) to debias a multi-class machine learning classifier's prediction of a person's income with respect to their race and gender. First, it is shown that the multi-class classifier, which uses California census data, is biased through a quantitative score and high misclassification rates. Next, taking inspiration from classical GAN architecture, two neural networks are created: a predictive network that takes in a person's features, excluding race and gender, to predict their income; and an adversarial network that infers the person's race and gender from the predictive network. To prove the generalizability of the GAN, the GAN is used to debias a Natural Language Processing (NLP) task: a word vector association task trained on 1,000 random Wikipedia articles. A decrease in bias is observed when the GAN is applied to the multi-class classifier and the word vector association task. The classifier, which originally had p-% of 39% for race and 30% for gender, increased to 76% for race and 82% for gender after applying the GAN. It has been shown that artificial intelligence, more specifically GANs, can be used to decrease the bias in machine learning algorithm outputs; the algorithm can be easily applied to real-world situations such as hiring employees or approving loans.

KEYWORDS: Robotics and Intelligent Machines; Machine Learning; Gender Bias; Racial Bias; Neural Network; Generative Adversarial Network (GAN); Natural Language Processing (NLP).

Introduction

Why Eliminating Bias Is Important:

For the purposes of this paper, bias is defined as a condition where an algorithm when queried for data returns one result at a higher rate than another, based on its reading of 'sensitive attributes' such as race and gender. In other words, an algorithm is biased if it unfairly prefers some groups of data over others. The GAN algorithm proposed in this research debiases existing algorithms that prioritize men over women, and White people over people who are Black, Asian, or Native American. A more detailed and quantifiable definition of fairness can be found in the "Meaning of Fairness" section of this paper.

Machine learning consists of algorithms that are exposed to training data which then improve their abilities through experience. Unfortunately, biases are often associated with this process. For example, the word doctor may typically be associated with a man because of the bias in the inputted dataset. As machine learning is trained on data, bias present in any given data is paralleled in the machine learning algorithm.

There are disparities in life that cannot be debiased: for example, women are more likely to live longer than men. A bias exists wherever there is a disparity in results that does not necessarily reflect reality. Disparities in data lead to biased results from the algorithms that are trained on said data. For instance, a woman may be statistically more likely to be a nurse than a man, but if this fact prevents men from becoming nurses it is an example of bias. When this fact does not prevent men from becoming nurses, it is an example of disparity. A biased result can easily occur in machine learning. Considering the prior example, here is a scenario that could occur. First, an algorithm takes data containing a disparity—such as a dataset showing that more women are nurses than men—and trains itself to implicitly recognize that women are better nurses than men because of that original disparity. Over time, the algorithm perpetuates its skewed understanding by recommending that a company should hire more women than men for nursing roles. This project does not create a solution for algorithms that are biased by design. Rather, this approach attempts to decrease the bias in machine learning algorithms that are created because they perpetuate any unjust disparities that exist in the data they are given.

Ideally, unbiased data would be accessible. However, this is not always possible as equitable data are not easily obtainable. Current approaches include collecting more data. However, this is expensive, hard to implement, and time-consuming. Thus, contemporary research has found that most industry leaders are not taking this approach.

As artificial intelligence becomes a pervasive tool in many aspects of daily life, assisting in operations from job recruitment to loan approval and law enforcement, it becomes increasingly important to understand and combat the tendency of machine learning algorithms to reproduce any bias contained in the data they use. Corporations such as Amazon, Hilton Worldwide Holdings, and Goldman Sachs have expressed their intention to use machine learning algorithms to automate aspects of their hiring processes. Amazon has experimented with using an algorithm to filter a pool of applicants, determine the top candidates according to parameters, and then send only the selected applications to the human resources department.[1]

DOI: 10.36838/v3i6.5

Amazon's experimental hiring tool was revealed to be downgrading female applicants because it was recreating disparities in its dataset. Men had traditionally been more successful than women at obtaining jobs at the company, a reality attributable to the long-standing gender gap in the technology industry, so the algorithm had taught itself that male candidates were more likely to be successful applicants than female candidates. As a result, the algorithm began downgrading the resumes of women such that "it penalized resumes that included the word 'women's,' as in 'women's chess club captain.' And it downgraded graduates of two all-women's colleges."[1] Although Amazon edited the algorithm so as to make it "neutral" to these terms, critics remain concerned that these changes cannot guarantee that the algorithm will not teach itself to develop other types of prejudice based on disparities in the data it uses.[1]

Framework:

For the general process of machine learning, data are first inputted into the neural network. Processing then takes place in the hidden layers through connections. Patterns are found and weights are assigned to each pattern, depending on how important that pattern seems. Finally, the hidden layers link to the output layer, where the outputs are retrieved.

A Generative Adversarial Network, or GAN, is a specific type of machine learning where two neural networks, a generator and a discriminator, compete with each other in a zero-sum game. Typically, the generator creates fake data and the discriminator attempts to classify the data. Then, the discriminator will adjust either the generator or itself in an attempt to improve its accuracy. This process, also known as adversarial learning, continues until the discriminator is not able to distinguish between the real and the generated data.

Natural Language Processing, or NLP, is a subfield of linguistics, computer science, and artificial intelligence that discovers interactions between computers and human language. It focuses on computer processing and analyzing large amounts of natural language data. NLP uses word vectors, or geometrical representations of words, to calculate word similarities.

Related Works:

Buolamwini *et al*. (2018) highlight how machine learning algorithms, particularly the facial recognition software created by Microsoft, IBM, and Face++, may discriminate against users based on attributes such as race and gender.[2] While a program's algorithm itself may not be biased, when data used to train the algorithm contains disparities, these differences can be replicated in the results, thus producing biased outcomes. Buolamwini *et al*. show how the data inputted into IBM, Microsoft, and Face++'s facial analysis algorithms were disproportionately composed of lighter-skinned subjects and contained more male than female data points. As a result, these algorithms often misclassified darker-skinned females, with error rates up to 34.7%. In contrast, the maximum error rate for lighter-skinned males was only 0.8%.[2] In their study, Buolamwini *et al*. provide real-world examples of the grave consequences that result from datasets that do not fairly represent all populations, warning that biased facial recognition software could threaten to result in unfair convictions and influence health care outcomes. The paper concludes by describing how a solution is urgently needed in order to prevent cultural biases from being perpetuated by machine learning algorithms, especially when AI is "rapidly infiltrating every aspect of society."[2]

Weber *et al*. (2020) discusses how artificial intelligence used to determine loan eligibility does not produce fair results due to its reliance on historical data which are " irreparably poisoned by structural and cultural racism, past and present."[3] Although redlining was outlawed in the 1968 Fair Housing Act, machine learning algorithms continue to use these maps to determine loan eligibility, thus discriminating against people of color such that "today, tens of millions of creditworthy prospective borrowers are excluded from credit access or charged with 'racial premiums' because their creditworthiness is determined by models failing to disambiguate historical oppression and present merit."[3,4] Weber *et al*.'s research sheds light on the drastic changes that need to be made to machine learning algorithms if they are used to determine loans. As the study states, the Black Lives Matter movement has demonstrated that the United States continues to discriminate based on race, and these algorithms are part of the problem.[3]

Recently, there have been several attempts to use artificial intelligence in order to solve the biases that machine learning algorithms reproduce as a result of the disparities in their collected data.

Zhang *et al*. (2018) used adversarial learning, more specifically logistic regression, in an attempt to eliminate the bias in results produced by data including gender disparities.[5] Logistic regression is a statistical model that determines if a variable has an effect on the output. Zhang *et al*. established three variables: X, y, and Z, where X is the input, which is census data in this example; y is the prediction, such as an income bracket; and Z is the gender or zip code. Zhang *et al*. did not include race in their study. The objective of Zhang *et al*.'s algorithm is to maximize the ability to predict y, while minimizing the ability to predict Z, given an input X. Thus, logistic regression is used in order to ensure that gender does not have an effect on the occupation output. The Zhang *et al*. study also sought to reduce the bias in word embeddings. Through an application of Bolukbasi *e t a l*.'s methodology, Zhang *et al*. were able to reduce the number of biased analogies that they originally observed. Although, some basic gendered analogies such as man : woman :: he : she were still preserved.[5] It is important to note that while Zhang *et al*.'s research developed a promising approach, it still produced a biased result.

Beutel *et al*. (2018) used adversarial learning to remove certain sensitive attributes in order to not expose the machine learning algorithm to these attributes.[6] As a result, the algorithm was able to provide its output, whether that be a recommendation to hire someone or to approve a loan, without being influenced by characteristics such as one's race, gender, or ethnicity.[6] However, even if the algorithm was not exposed to these sensitive attributes, it still produced a biased result due to the patterns in the data.

DOI: 10.36838/v3i6.5

While Tonk (2018) acknowledges that more papers are being published on the issue of fairness and bias in machine learning, he states that strides still need to be made in the field. Tonk describes how machine learning models are increasingly being used in everyday decision-making processes and states that the fairness of the results produced by these algorithms must be prioritized so as to "mitigate emergent discrimination" in machine learning models.[7] Tonk's study trained a model making income level predictions and then used adversarial learning to make the output fairer. This work was inspired by Louppe *et al.*'s 2017 paper, which shows how adversarial networks can be used when the training data do not accurately represent the world as a whole.[7,8] Tonk's approach used adversarial learning to attempt to debias a linear binary classifier's ability to predict whether a person earned a salary of more than $50,000 without basing those predictions on biased assessments such as race and gender.[7] It is important to note that Tonk did not consider other racial groups besides Black and White in his study. Tonk used a p-% metric value to measure the bias of the binary classifier's result before and after an adversarial learning algorithm was applied; he noticed a decrease in bias of more than 40% for both race and gender. Tonk's work takes the conclusions made in Beutel *et al*. further, stating that although eliminating the adversarial network's ability to see characteristics like race or gender does improve the biased results, it does not correct them entirely.[7]

Bolukbasi *et al*. (2016) focused on the bias created when word vectors are trained on Google News articles.[9] Word embeddings are described as vectors that represent words, acting as a dictionary which attempts to interpret the meaning of a word for computer programs. The Google News articles used in the study contain immense amounts of discriminatory perspectives and associations, which were in turn amplified by the machine learning algorithms that used them as data. To attempt to debias the results, Bolukbasi *et al*. modified the embeddings to remove gendered associations between words, such as woman and nurse, while also preserving appropriate associations, such as between woman and queen.[9] In order to do this, two steps were used: "The first step, called Identify gender subspace, is to identify a direction (or, more generally, a subspace) of the embedding that captures the bias. For the second step, we define two options: Neutralize and Equalize or Soften. Neutralize ensures that gender neutral words are zero in the gender subspace. Equalize perfectly equalizes sets of words outside the subspace and thereby enforces the property that any neutral word is equidistant to all words in each equality set."[9] Bolukbasi *et al*.'s machine learning model resulted in a 13% decrease in gender-biased results.[9] However, their approach is very specific and not easily applicable to other situations as it can only be applied to word vectors.

In this paper, inspiration is drawn from Zhang *et al*., and the methods and p-% metric used to measure bias are similar to Tonk's study; however, this research is arrived at independently and the approaches are very different. This paper goes beyond Zhang *et al*. and Tonk's binary class classifier, as a machine learning multi-class classifier is used to expand the scope of debiasing and create a GAN algorithm that further decreases biases. The proposed GAN algorithm is changed from three tiers to a two-layer neural network with many of the parameters altered. Subsequently, the proposed GAN algorithm is then applied to an NLP word vector association task in an attempt to debias word associations so as to demonstrate the generalizability of the debiasing GAN algorithm to a different dataset and underlying algorithm.

Research Question and Hypothesis:

In past research, GANs have been used to increase algorithmic fairness by creating or collecting new data that do not contain disparities. Instead of taking this expensive approach of collecting new data, inspiration is drawn to create two neural networks.[10]

This paper explores whether or not implementing a GAN algorithm to create a classifier-adversary neural network structure will be able to decrease the biased results of a multi-class classifier according to the p-% rule. To prove the generalizability of this approach, the GAN algorithm is also applied to decrease the biased results of an NLP word vector association task.

It is expected that this GAN algorithm will be able to decrease the biased results of both machine learning tasks as it takes advantage of the zero-sum-game nature of adversarial learning in order to improve the classification accuracy and word vector associations while minimizing the ability to predict certain attributes such as race and gender. When applied, this algorithm should ensure that companies are not able to detect one's race or gender when determining whether or not they hire a candidate.

Methods

Setup and Dataset for a GAN Application to a Multi-Class Classifier:

Taking inspiration from Bolukbasi *et al*. (2016), Beutel *et al*. (2017), Zhang *et al*. (2018), and Tonk (2018), training of the machine learning algorithm and evaluation of its fairness is completed using the University of California, Irvine (UCI) census data.[5-7,9,11] As the ultimate goal is to create an algorithm that can be used by financial institutions and the government, it is appropriate that publicly available census data are chosen for this study, which represents 30,940 individuals. Details on the features of this dataset are included in Table 1.

To establish a baseline before implementing the GAN, the UCI census dataset was used to train and test the ability of a linear multi-class classifier to predict whether or not a given person in the dataset earned more than a $50,000 salary per year. First, the dataset is split in a 70:20:10 ratio into a training set, testing set, and validation set.

In order to do this, the data are divided into three sets: features, targets, and sensitive attributes. The characteristics, represented by X, contain attributes such as age, education, and marital status, which the multi-class classifier uses to make predictions. The targets, represented by y, include the labels that the classification algorithm outputs: whether or not one's income is above $50,000 a year. The sensitive attributes, represented by Z, contain the multi-class attributes that need to be represented fairly: Black, White, Asian/Pacific Islander, and American Indian for race; and male and female for gender. The sensitive attributes are not in the set of machine learning

DOI: 10.36838/v3i6.5

features, meaning the model is not exposed to race and gender while initially training the machine learning classifier.

Table 1: Features of the UCI census dataset.[11]

Feature	Description
age	Individual's age
capital_gain	Capital gains recorded
capital_loss	Capital losses recorded
education_num	Highest educational degree received (in numerical form)
fnlwgt	Total number of people census takers believe that the observation represents
hours_per_week	Hours worked by the individual per week
education	Highest level of education achieved
income	Boolean: whether individual has income > $50,000/year
marital_status	Marital status
native_country	Birth country
occupation	Occupation
race	Black, White, Asian/Pacific Islander, American Indian, Other
relationship	Wife, Husband, Own-child, Not-in-family, Other-relative, Unmarried
gender	Male, Female
workclass	Employer type

Due to a lack of data, certain omissions had to be made: only male and female could be used for gender, and Hispanic and other races could not be included.

Training the Basic Income Level Predictor:

In order to predict the probability of one's income being over $50,000 a year, NumPy, a Python machine learning library, is used to fit a two-layer neural network.[12] The Leaky ReLU function is used as an activation function, as it has become a standard in machine learning in recent years, and stochastic gradient descent is used to train the machine learning classifier.[13] Additionally, the learning rate for the neural network is adjusted with an exponential learning rate schedule as optimization proceeds. After each training iteration, the learning rate is reduced by multiplying it by a decay rate of 0.95. See Figure 1 of the Appendix for the Python code written to train the neural network.

Developing a GAN Algorithm:

The machine learning classifier did not have access to the defined sensitive attributes, but it still produced biased results. The classifier behaved in this manner due to implicit biases associated with other input features such as age, occupation, or education.

The GAN algorithm described by Goodfellow *et al*. consists of two neutral networks: a generative model and an adversarial classifier.[14] Both compete in a zero-sum game in order to create simulated data that are representative of the data that already exist. The generative model, or discriminator, creates the data while the adversarial classifier, or generator, tries to distinguish whether or not they are real data. As both networks are being trained at the same time, the algorithm continues to improve.[14]

Taking inspiration from Goodfellow *et al*., the algorithm explored in this paper applies a GAN algorithm in addition to the classifier to constrain the model, so it is forced into making fair predictions. The new model tries to enforce the pivotal property on the predictive model capable of predicting income levels so that sensitive attributes do not affect the predictions. By taking the sensitive attributes as nuisance parameters, those on which the neural network is not explicitly trained, predictions can be enforced that are independent of race and gender—leading to fair predictions.

The algorithm proposed in this work keeps the GAN functionality of the discriminator network, which is to predict whether or not one earns a salary of more than $50,000 a year based on the X input. However, this work repurposes the functionality of the generator from creating simulated data, as is done in Louppe *et al*.'s 2017 paper, predicting the sensitive attributes Z from the predicted y without direct access to X.[8]

The GAN can be thought of as a back-and-forth process of the generator trying to fool the discriminator and the discriminator trying to correctly classify whether or not one earns a salary of more than $50,000 a year. This is a minimax game captured by Eq. 1 below, where $Z \sim p(z)$ are random noise samples, $G(z)$ are the outputs of the generator, and D is the output of the discriminator.

(Eq.1)

$$minimize_G maximize_D \; E_{x \sim p_{data}}[\log D(x)] + E_{z \sim p(z)}[\log (1 - D(G(z)))]$$

See Figure 2 of the Appendix which shows the code implementation of the GAN loss function (Eq. 1).

The loss function captures the zero-sum nature of the algorithm where the generative network learns to minimize its predictive losses while maximizing that of the adversarial network.

Generator and Discriminator Architecture and Training the GAN:

To optimize Eq. 1, gradient descent steps on the objective for G and gradient ascent steps for the objective for D are taken, alternating between the two. At a high level, the goal is to update the generator to minimize the likelihood of the discriminator correctly classifying the given person's race and update the discriminator to maximize the probability of the discriminator predicting whether or not the given person makes a salary of more than $50,000 a year. All code implemented for the GAN algorithm is done in Python TensorFlow.[15]

Now that the objective function for the GAN has been clearly defined, the discriminator network can be built. The details for the layers of the discriminator are: a fully connected layer with input size 784 and output size 256, Leaky ReLU activation function, a fully connected layer with input size 256 and output size 256, Leaky ReLU activation function, and a fully connected layer with input size 256 and output size 2. See Figure 3 of the Appendix for the Python code written for the discriminator architecture.

The details for the layers of the generator network are as follows: a fully connected layer with input size 1024, Leaky ReLU activation function, a fully connected layer with input size 1024, Leaky ReLU activation function, a fully connected layer with input size 784, and a hyperbolic tangent activation function to restrict all classification outputs for the sensitive attributes to a range of [-1, 1]. See Figure 4 of the Appendix for the Python code written for the generator architecture.

With the discriminator and generator networks defined, the GAN can be trained. First, the generator is trained on the full dataset. Then, the discriminator is pre-trained on the full dataset without exposure to the generator's outputs. Finally, the generator and discriminator are trained simultaneously over 250 iterations.

Meaning of Fairness:

In order to quantitatively determine the success of the algorithm, several metrics are used. Some examples include demographic parity, which tests if two variables are independent; equality of odds, which tests if two variables are independent of each other and is defined in Eqs. 1 and 2; and equality of opportunity, which tests if two variables are conditionally independent given one of them.[5]

In this work, inspiration is drawn from the U.S. Equal Employment Opportunity Commission's (EEOC) 80% rule.[7] The 80% rule looks at the ratio of two groups being compared, whether that be Black and White for race or male and female for gender. In their 2014 paper, Zafar *et al*. demonstrated how a more generic version of this rule, called the p-% rule, can be used to quantify fairness of a classifier.[16]

The p-% rule states that the ratio between the probability of a positive outcome given the sensitive attribute—race and gender—being true and the same probability given the sensitive attribute being false is no less than the ratio p:100.[6] Thus, when a classifier is completely fair it will satisfy a 100% rule. In contrast, when it is completely unfair it satisfies a 0% rule. In this work, three p-% rule metrics are calculated for race and use White as a standard comparison for racial bias. More precisely, for race, the p-% rule is calculated for White and Black, White and Asian/Pacific Islander, and White and American Indian. For gender, the p-% rule is calculated for male and female. The classifier in this paper will be determined fair if it satisfies at least an 80% rule, following the EEOC.[6]

Given the sensitive attributes, the p-% with respect to race for this model is calculated using the equation below in order to quantitatively see how fair the income level predictor proves. For Eq. 2, y =1 indicates that the machine learning classifier predicts that the given person earns a salary of more than $50,000, and Z can be substituted with the races under consideration in this work: Black, Asian/Pacific Islander, and American Indian.[6]

(Eq.2)[16]

$$\min\left(\frac{P(y=1|z=white)}{P(y=1|z=other\ race)}, \frac{P(y=1|z=other\ race)}{P(y=1|z=white)}\right) \geq \frac{p}{100}$$

The p-% equation (Eq. 3) for gender is similar, but since only male and female are being considered as gender classifications, the value of Z in this equation does not change.

(Eq.3)[16]

$$\min\left(\frac{P(y=1|z=male)}{P(y=1|z=female)}, \frac{P(y=1|z=female)}{P(y=1|z=male)}\right) \geq \frac{p}{100}$$

Setup and Dataset for a GAN Application to an NLP Word Embedding Task:

It is common in NLP to represent words as vectors so that qualitative relationships between words can be quantified. As described in Bolukbasi *et al*.'s 2016 study, word embeddings are a framework to represent text as vectors which can be used as input data for machine learning and NLP tasks.[9] Word embeddings typically contain data from written articles. A method of calculating a word vector would be to count the number of occurrences of a specific word in a document over all of the documents in the dataset. Each document is a dimension of the vector. The vectors allow computers to understand the meaning of a word. Then, with the use of these vectors, the similarities between words can be determined and used for numerous applications. While word embeddings can be helpful for determining the similarity between words because they synthesize existing information in a seemingly unbiased way, they also perpetuate any disparities and biases contained in the documents from which they are derived.

This research attempts to eliminate the gender bias that is produced when using word embeddings as data for machine learning software. To create the word embeddings, 1,000 randomly selected Wikipedia articles are collected and word vectors are created for all of the words in the dataset. This study uses Wikipedia articles because they are regularly updated, in contrast to articles from sources such as Google News, as used by Bolukbasi *et al*., which can become outdated. Similar to Bolukbasi *et al*.'s determination of bias in Google News articles, this study found that Wikipedia contributors reproduce their own biases in the information they write.[9] See Figure 5 of the Appendix for this paper's method of creating word vectors using Python code.

To add more complexity, dimensions can also be added that include the instances the word appears next to another word, which has been implemented in Figure 6 of the Appendix.

Once the vectors are established, statistical methods can be used to calculate the similarity between vectors. In this work, the cosine similarity is used as it is most common for text similarity. Next, a similarity score for the cosine similarity is calculated; a number between 0 and 1 will be returned, 1 meaning they are very similar and 0 meaning they are not similar. For example, in Figure A below, the words closer to the x-axis are most similar to the word woman.[9]

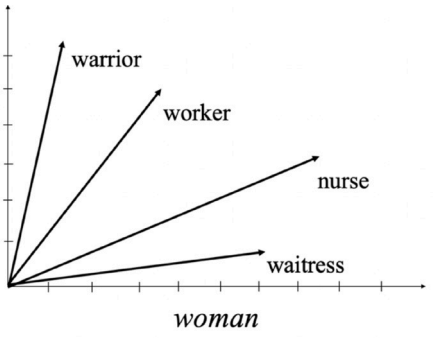

Figure A: Geometric visualization of word vectors and cosine similarity.

Eq. 4 shows the cosine similarity formula, which is converted to code in Figure 7 of the Appendix.

(Eq.4)[9]

$$\text{Cosine}(\mathbf{v},\mathbf{w}) = \frac{\mathbf{v}\bullet\mathbf{w}}{|\mathbf{v}||\mathbf{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

DOI: 10.36838/v3i6.5

The ultimate goal of this NLP task is to find the five words most similar to man and woman. In order to do this, the word vector dictionary created earlier was searched, and the five words that have the smallest cosine similarity to man and woman respectively are returned. Figure 8 of the Appendix shows the Python code implementation of this process.

Applying the GAN Algorithm to the NLP Word Embedding Task:

In order to apply the GAN algorithm to the NLP word embedding task, two steps are needed. First, the adversarial network calculates the cosine similarities between all possible words with respect to man and woman. This provides a method for defining the protected gender variable. Then, the network outputs five words that have the highest cosine similarity, and are thus the least similar to man and woman. These words are defined as the "bias subspace."

Then, the discriminator network projects all the words in the "bias subspace" to a word vector that is perpendicular to the word vectors for man and woman. It is expected that the discriminator network will learn that an "unbiased subspace" would contain vectors that should be associated with man and woman. Next, the new cosine similarities between the set of projected word vectors in the "unbiased subspace" to man and woman can be calculated. The five words in the "unbiased subspace" that have the smallest cosine similarity to man and woman are chosen.

Since the discriminator network now associates words that are considered to be opposite to the meaning of man and woman in a biased setting, it is associating unbiased words to man and woman.

Results and Discussion

Results and Discussion for Classification Task:

The initial accuracies for the multi-class classifier were 82% in classifying race and 80.2% in classifying gender. Furthermore, the p-% for minority races compared to White were as follows: 28% for Black, 63% for Asians, and 25% for Native American. The p-% for gender was 30%. From this research's definition of the p-% value, the multi-class classifier is less likely to predict someone who is Black, Asian, or Native American to make an income of over $50,000 a year in comparison to someone who is White. Similarly, the classifier is also less likely to predict that a woman will make an income of over $50,000 a year in comparison to a man, as can be seen by the low p-% value.

After the GAN algorithm is applied to the classifier, the classification accuracies remained approximately the same. Furthermore, the prediction accuracies are 76% in classifying race and 78% in classifying gender. The p-% for minority races all significantly increased and are now as follows: 78% for Black, 85% for Asians, and 65% for Native American. The p-% for gender also rose and is now 82%. The percentages are all listed in Table 2 below.

The p-% for gender now is above the 80% threshold that was deemed to be fair earlier in this paper. While the p-% for Black and Native Americans does not meet the 80% threshold, there is still a significant increase in the p-% value.

Table 2: Results of debiasing the multi-class classifier.

	Accuracy		p-% for Race			p-% for Gender
	Race	*Gender*	*Black*	*Asian*	*Native American*	
Before	82%	80.2%	28%	63%	25%	30%
After	76%	78%	78%	85%	65%	82%

It can be predicted that the machine learning classifier accuracy in this paper decreased as a result of the data. The original p-% is low as a result of disparities in the data. Now that the results are less biased, the accuracy in terms of the original dataset slightly decreased. However, the decrease of the p-% value is miniscule and can be attributed to difficulties in the ability of the algorithm to converge to global minima instead of local minima.

The most similar attempt to combat bias in the output of machine learning tasks was performed by Zhang *et al.* in "Mitigating Unwanted Biases with Adversarial Learning."[5] This paper uses the same UCI census data and adversarial learning idea as Zhang *et al.*, and thus the two studies can be easily compared.[5,11] To begin with, the Zhang *et al.* paper started with an 86.0% accuracy, and dropped to an 84.5% accuracy.[5] In this work, there is no noticeable change in the classification accuracy. The Zhang *et al.* paper only considered gender debiasing, and not race, so the p-% results will only be compared with this study for gender. In addition, Zhang *et al.* used a different optimization, Python library, and measurement of bias.[5] Table 3 shows a comparison between the work discussed in this paper and Zhang *et al.*'s work with and without a debiasing algorithm applied. The accuracy column in Table 3 represents the machine learning classifier's accuracy.

Table 3: : P-% value in the current work as compared to Zhang *et al.* with and without GAN applied.[5]

	Accuracy		Gender Bias (p-% value)	
	Without	*With*	*Without*	*With*
Zhang et al. (2018)	86.0%	84.5%	25%	62%
Current work	82%	80.2%	30%	82%

The p-% value for the Zhang *et al.* paper increased by 37 percentage points, while this current work increased by 52 percentage points.[5] This increase in difference between the p-% values for before and after the adversarial networks shows the method described in this paper is more successful at eliminating the output's bias. In addition, this method is more easily applicable as it has also been implemented on both race and gender, while the Zhang *et al.* paper only explored gender.[5]

Tonk's work also included both race and gender; however, it is limited to only Black and White racial groups. Tonk also uses Keras, a binary classifier, and a different optimization equation.[7]

Results and Discussion for Word Embedding Task:

Although the results for the word embedding task are not quantifiable, from a subjective view the occupations provided for each gender perpetuate fewer of the disparities contained in the original data, as can be seen in the results of Table 4. For example, before applying the GAN, the occupations for women were associated with positions of lesser power or skill, such as a nurse or waitress. After applying the GAN, women are presented in positions of relative power such as a surgeon or pediatrician. In the future, it would be interesting to create

a numerical score in order to quantifiably measure the decrease in bias in the outputs instead of having to rely on a subjective point of view.

Table 4: Word embeddings before and after debiasing.

Man		Woman	
Before	*After*	*Before*	*After*
warrior	fiancé	waitress	pediatrician
lawyer	dentist	teacher	dentist
teacher	actor	nurse	surgeon
actor	doctor	maid	teacher
worker	hospital	worker	house

Conclusion

Taking inspiration from a GAN, the algorithm developed in this study maximized the ability to predict *y* (whether or not one's income is above $50,000 a year), while minimizing the ability to predict *Z* (the sensitive attributes, such as race and gender), given an input *X* (the dataset). A multi-class classifier is used as a baseline metric to compare to the algorithm's performance. In addition, the GAN algorithm is used in conjunction with the classifier so as to eliminate the bias in its results.

In order to quantitatively measure the fairness of the algorithm, the p-% rule, in addition to the EEOC's 80% rule, is used. In this study, the algorithm is deemed fair if the p-% is greater than 80%. The classifier, which originally had p-% of 39% for race and 30% for gender, increased to 76% for race and 82% for gender with the use of the GAN algorithm created in this study. While the p-% for race is not deemed fair, it has a significant increase from before the GAN was applied. In an effort to make the p-% fair, a more diverse dataset could be used in addition to the GAN algorithm in the future.

An extra layer of complexity is added by extending the GAN algorithm application to the NLP word embedding task. This is done in order to prove the generalizability of the GAN algorithm. As a result of this approach, there is an observed subjective decrease in biased results. This means that this GAN algorithm can be generalizable for other forms of classification bias as it is successful for more than one sensitive attribute.

In the future, this GAN algorithm can be used with resume data in order to minimize the influence of sensitive attributes, such as race or gender, on who is hired. Similarly, the algorithm can reduce the correlation between sensitive attributes and biased results for many other purposes.

Acknowledgements

I would like to thank Ms. Amber Yang of Stanford University for being my mentor and helping to guide me throughout the entire process of conducting my research and drawing conclusions. In addition, I would like to thank Professor Julia Hirschberg of Columbia University, PhD University of Pennsylvania; Professor K. David Harrison of Swarthmore College, PhD Yale University; and Professor Susan Behrens of Marymount Manhattan College, PhD Brown University, for reviewing this paper for publication and for providing encouragement and mentorship as I deepened my understanding of computer science, linguistics, and computational linguistics. Finally, I would like to thank my teachers at The Spence School: Mr. Scott Godsen for inspiring my interest in the sciences, Mr. Michael Gold for fostering my love of mathematics, and Ms. Julie Abbruscato for encouraging me to take risks and pursue my passion for linguistics outside of the high school classroom.

References

1. Dastin, J. Amazon Scraps Secret AI Recruiting Tool That Showed Bias Against Women. Reuters, Oct 10, 2018. https://www.reuters.com/article/us-amazon-com-jobs-automation-insight/amazon-scraps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK08G (accessed March 15, 2020).
2. Buolamwini, J.; Gebru, T. Gender shades: Intersectional Accuracy Disparities in Commercial Gender Classification. *Proceedings of Machine Learning Research* 2018, 81, 1-15. http://proceedings.mlr.press/v81/buolamwini18a/buolamwini18a.pdf (accessed March 16, 2020).
3. Weber, M.; Yurochkin, M.; Botros, S.; Markov, V. Black Loans Matter: Distributionally Robust Fairness for Fighting Subgroup Discrimination, 34th Conference on Neural Information Processing Systems (NeurIPS), Fair AI in Finance Workshop. Nov 30, 2020, arXiv:2012.01193v1 [cs.CY], arXiv.org e-Print archive. https://arxiv.org/pdf/2012.01193.pdf (accessed Nov 30, 2020).
4. Weber, M.; Yurochkin, M.; Botros, S.; Markov, V. Black Loans Matter: Fighting Bias for AI Fairness in Lending. MIT-IBM Watson AI Lab, Nov 27, 2020. https://mitibmwatsonailab.mit.edu/research/blog/black-loans-matter-fighting-bias-for-ai-fairness-in-lending/ (accessed Nov 30, 2020).
5. Zhang, B. H.; Lemoine, B.; Mitchell, M. Mitigating Unwanted Biaseswith Adversarial Learning. Jan 22, 2018, arXiv:1801.07593 [cs.LG], arXiv.org e-Print archive. https://arxiv.org/pdf/1801.07593.pdf (accessed March 20, 2020).
6. Beutel, A.; Chen, J.; Zhao, Z.; Chi, E. H. Data Decisions and Theoretical Implications When Adversarially Learning Fair Representations. July 7, 2017, arXiv:1707.00075 [cs.LG], arXiv.org e-Print archive. https://arxiv.org/pdf/1707.00075.pdf (accessed March 30, 2020).
7. Tonk, S. Towards Fairness in ML with Adversarial Networks. GoDataDriven, Apr 27, 2018. https://godatadriven.com/blog/towards-fairness-in-ml-with-adversarial-networks/ (accessed April 3, 2020).
8. Louppe, G.; Kagan, M.; Cranmer, K. Learning to Pivot with Adversarial Networks. 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, 2017. https://papers.nips.cc/paper/6699-learning-to-pivot-with-adversarial-networks.pdf (accessed April 1, 2020).
9. Bolukbasi, T.; Chang, K.-W.; Zou, J.; Saligrama, V.; Kalai, A. Man Is to Computer Programmer as Woman Is to Homemaker? Debiasing Word Embeddings. July 21, 2016, arXiv:1607.06520v1 [cs.CL], arXiv.org e-Print archive. https://arxiv.org/abs/1607.06520.pdf (accessed March 16, 2020).
10. Xu, D.; Yuan, S.; Zhang, L.; Wu, X. FairGAN: Fairness-aware Generative Adversarial Networks. May 28, 2018, arXiv:1805.12202 [cs.LG], arXiv.org e-Print archive. https://arxiv.org/pdf/1805.11202.pdf (accessed March 27, 2020).
11. Dua, D.; Graff, C. Adult Data Set. 2019, UCI Machine Learning Repository. http://archive.ics.uci.edu/ml/datasets/Adult (accessed March 28, 2020).
12. *NumPy*, version 1.19.0; NumPy: 2005, https://numpy.org/ (accessed May 3, 2020).
13. Nwankpa, C.; Ijomah, W.; Gachagan, A.; Marshall, S. Activation Functions: Comparison of Trends in Practice and Research for Deep Learning. Nov 8, 2018, arXiv:1811.03378 [cs.LG], arXiv.org e-Print archive. https://arxiv.org/pdf/1811.03378.pdf (accessed March 30, 2020).
14. Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.;

DOI: 10.36838/v3i6.5

Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Nets. Jun 10, 2014, arXiV:1406.2661 [stat.ML], arXiv.org e-Print archive. https://arxiv.org/pdf/1406.2661.pdf (accessed April 10, 2020).
15. *TensorFlow*, version 2.4.0; GitHub. https://www.tensorflow.org/ (accessed May 15, 2020).
16. Zafar, M. B.; Valera, I.; Rodriguez, M. G.; Gummadi, K. P. Fairness Constraints: Mechanisms for Fair Classification. Mar 23, 2017, arXiv:1507.05259 [stat.ML], arXiv.org e-Print archive. https://arxiv.org/pdf/1507.05259.pdf (accessed April 1, 2020).

Author

Isabella Mandis is in Grade 12 at The Spence School in New York City. In 2020, Isabella founded www.languagepreservationinitiative.com, which raises awareness of the importance of sustaining endangered languages and showcases her research in linguistics. Isabella is currently working to debias machine learning algorithms used for Natural Language Processing.

Appendix

```
input_size = 32 * 32 * 3
hidden_size = 50
num_classes = 10
net = TwoLayerNet(input_size, hidden_size, num_classes)

# Train the network
stats = net.train(X_train, y_train, X_val, y_val,
            num_iters=1000, batch_size=200,
            learning_rate=1e-4, learning_rate_decay=0.95,
            reg=0.25, verbose=True)

# Predict on the validation set
val_acc = (net.predict(X_val) == y_val).mean()
print('Validation accuracy: ', val_acc)
```

Figure 1: Python code for training a two-layer neural network.

```
def gan_loss(logits_real, logits_fake):
    D_loss = None
    G_loss = None

    Dx = tf.nn.sigmoid_cross_entropy_with_logits(logits=logits_real, labels=tf.ones_like(logits_real))
    DGx = tf.nn.sigmoid_cross_entropy_with_logits(logits=logits_fake, labels=tf.zeros_like(logits_fake))
    Gx = tf.nn.sigmoid_cross_entropy_with_logits(logits=logits_fake, labels=tf.ones_like(logits_fake))
    D_loss = tf.reduce_mean(Dx) + tf.reduce_mean(DGx)
    G_loss = tf.reduce_mean(Gx)
    return D_loss, G_loss
```

Figure 2: Python code for GAN loss function.

```
def discriminator(x):
    with tf.variable_scope("discriminator"):
        fc1 = tf.layers.dense(x, 256)
        lr = leaky_relu(fc)
        fc2 = tf.layers.dense(lr, 256)
        lr2 = leaky_relu(fc2)
        fc3 = tf.layers.dense(lr2, 2)
        return scores
```

Figure 3: Python code written for the discriminator architecture.

```
def generator(z):
    with tf.variable_scope("generator"):
        fc1 = tf.layers.dense(z, 1024, activation=tf.nn.relu)
        fc2 = tf.layers.dense(fc1, 1024, activation=tf.nn.relu)
        output = tf.layers.dense(fc2, 784, activation=tf.nn.tanh)
        return output
```

Figure 4: Python code for generator architecture.

```
def find_synonym(word, choices, embeddings):
    cos_sim = [(choice, 1 - cosine_similarity(embeddings[word], embeddings[choice])) for choice in choices]
    cos_dict = {choice: dist for choice, dist in cos_sim}
    answer = min(cos_dict, key = cos_dict.get)
```

Figure 5: Python code implementing cosine similarity to find closest word synonyms.

```
def get_embedding(s, embeddings, use_POS=False, POS_weights=None):
    embed = np.zeros(embeddings.vector_size)

    if use_POS == False:
        word_tokens = word_tokenize(s)
        vector_word_tokens = embeddings[word_tokens]
        for vector in vector_word_tokens:
            embed += vector_word_tokens

    if use_POS == True:
        word_tokens = word_tokenize(s)
        tagged_words = nltk.pos_tag(word_tokens)
        dict_tagged_words = dict(tagged_words)
        vector_word_tokens = embeddings[word_tokens]
        vector_word_dict = {word_tokens: dist for word_tokens, dist in vector_word_tokens}

    return embed
```

Figure 6: Python code for creating a dictionary of associated word embeddings.

```
def cosine_similarity(v1, v2):
    cosine_sim = np.dot(v1, v2)/(np.linalg.norm(v1)*np.linalg.norm(v2))
    return cosine_sim
```

Figure 7: Python code for measuring cosine similarity between word vectors.

```
def occupation_exploration(occupations, embeddings):
    top_man_occs = []
    top_woman_occs = []

    man_embed = embeddings['man']
    woman_embed = embeddings['woman']

    man_cos_sim = [(occupation, (cosine_similarity(embeddings['man'], embeddings[occupation]))) \
                   for occupation in occupations]
    man_cos_dict = {occupation: dist for occupation, dist in man_cos_sim}
    top_man_occs_dict = dict(heapq.nlargest(5, man_cos_dict.items(), key=itemgetter(1)))

    for key in top_man_occs_dict.keys():
        top_man_occs.append(key)

    woman_cos_sim = [(occupation, (cosine_similarity(embeddings['woman'], embeddings[occupation]))) \
                     for occupation in occupations]
    woman_cos_dict = {occupation: dist for occupation, dist in woman_cos_sim}
    top_woman_occs_dict = dict(heapq.nlargest(5, woman_cos_dict.items(), key=itemgetter(1)))

    for key in top_woman_occs_dict.keys():
        top_woman_occs.append(key)

    return top_man_occs, top_woman_occs
```

Figure 8: Python code for generating top words associated with man and woman.