Novel Neural Network Repair Methods for Data Privacy and Individual Fairness

by

Laura Graves

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Masters of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2021

© Laura Graves 2021

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Machine learning is increasingly becoming critical to the decisions that control our lives. As these predictive models advance toward ubiquity, the demand for models that are trustworthy, fair, and preserve privacy becomes paramount. Despite this, significant privacy, trust, fairness, and security risks exist that make these models untrustworthy. Deep neural networks are vulnerable to attacks that reveal private information about training instances and violate regulatory guidelines. Additionally, models can display biased behavior which is difficult to detect and mitigate. To address these pressing questions, I present two main streams of research that fall under the umbrella of model repair. In the first, deemed Amnesiac Machine Learning, I address the problem of privacy leaking through two *unlearning* algorithms that specifically remove learning from a subset of training data. I evaluate these algorithms on a novel testing suite consisting of data-leaking attacks. In the second, I present an automated system that detects algorithmic bias, isolates the features most responsible for that biased behavior, and performs model repair to mitigate that bias. In both scenarios the repaired models have similar performance to models trained from scratch for the desired purpose, while at the same time do not exhibit privacy leakage or biased behavior on real-world data sets.

Acknowledgements

I would like to thank the following mentors and friends:

First and foremost, Dr. Vijay Ganesh, who has provided invaluable mentorship and inspiration through my graduate studies. I am grateful for your patience and insistence on "bit-precise" understanding of concepts, even when my stubbornness made that difficult. You gave me a great amount of freedom in research direction, but always provided excellent perspective and no shortage of interesting project avenues. You have made me a stronger researcher as well as a stronger person.

Dr. Antonina Kolokolova, who took a chance on me as an undergrad and provided me with my first research experience. Antonina, your passion for computer science and understanding is infectious, and I deeply admire your drive to share learning.

Dr. Florian Kerschbaum, who introduced me to a wide world of security and privacy issues and gave me a solid foundation for understanding them.

Dr. Chris Srinivasa, for your leadership and direction with Borealis AI. You have a fantastic team and I am glad for the time spent with them.

Dr. Mark Crowley, for providing feedback and guidance on the content of this material.

Vineel Nagisetty, my friend and accomplice through undergrad and graduate studies. You deserve a very bright future, and I have no doubt you will accomplish incredible things.

Finally, I want to thank Dr. Elliot Chapple, my source of comfort and strength during my studies. My world is better for having you in it.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis broadly focuses on topics related to deciphering the puzzles that are deep neural networks (DNNs). DNNs learn and process data in an opaque way, and this thesis covers methods to work with models despite this inscrutability. This thesis is broadly positioned in the greater field of repairing models to ensure that they have some specific property in addition to high accuracy on training and test data, a field which loosely encompasses methods to improve machine learning security, model compression, robustness, regularization, fairness, privacy, or any other method where a model must be manipulated during or after training. In general, it is accepted that training a model to maximize performance is not necessarily optimal toward other parameters. These other parameters, whether measured by some provable bound or measured by a metric, can be optimized for as part of a dual-objective optimization.

This thesis deals with the domain of post-training model manipulation. More specifically, this work is centered on the notion of *model repair*, where changes are made to a trained model that alter it to be more suitable for some task. Model training is expensive, time consuming, and often requires significant hyperparameter configuration and data processing. The approaches presented here aim to perform repair in an architecture-agnostic way, maintain model performance as much as possible, and require significantly less computation time than training a model from scratch to match the desired properties.

Post-training model repair is of significant benefit to model holders who may not know exactly what the future holds for the applications they are using these models in. In this thesis, I discuss the data privacy regulations introduced by the European Union, and analyze the potential risk that these regulations present. In particular, if an entity trains a model using data from some individuals, they may have a legal responsibility to remove that data from their systems at an individual's request, including whatever information is contained in trained neural networks [43]. With no way of knowing who may invoke this removal request, the stability of systems using deep learning can be threatened, keeping the entities using those models in a limbo where any request can force them to retrain their models [21]. There exists a great need for efficient methods to protect individual privacy without compromising the integrity of systems.

Post-training repair can also allow for repairs to be done for goals that were not previously considered. Machine learning models can exhibit significantly unfair behavior [46], and often this behavior is difficult to detect before training. In this thesis I present a method of repairing models toward the goal of individual fairness, where unfair behavior with regard to a demographic group or other identifiable feature can be detected and mitigated. If the group we wish to protect is unclear at the beginning of the training cycle, approaches such as this give great control to model holders who may have trained a model that unintentionally makes unfair predictions. Even if fairness was not a concern at the beginning, these repairs can be made to a fully trained model.

Although these are two excellent uses for post-training repair methods, many other methods exist for tangential problems. The field of model compression aims to produce a model that is smaller and less complex using a previously trained model. The field of transfer learning is similarly positioned, and both can be viewed as methods that keep important information while replacing unimportant information. This high level concept of retaining the good parts of what has been learned and replacing the less desirable parts is a useful frame for thinking about these problems.

In this larger field, the concept of model repair in particular is an important emerging subfield. The increasing data and training time required by state-of-the-art models also increase the need for repair methods that don't involve expensive training from scratch. The methods presented in this thesis both aim to mitigate this by not requiring a brand new model to be trained from scratch in order to modify characteristics; in fact, both methods require nothing but a small amount of fine tuning after the repair method is performed. It is in this paradigm that these methods are situated, that the best model repair methods are ones that can do complex repair tasks in a straightforward and efficient way.

The problem I address in this thesis is the following:

Problem Statement: *is it possible to modify a trained model to satisfy some property such as fairness or unlearning in a way that maintains model performance and does not require retraining from scratch?* My contributions to this problem are twofold, and detailed in the following chapters.

1.1 Contributions

The main contributions are as follows:

Machine Unlearning: I address the problem commonly known as *machine unlearning*, that is, I develop two algorithmic approaches to removing specific subsets of learning from a trained neural network. The first, which I refer to as *unlearning*, is suitable for removing large subsets of learning. The second, which I deem *amnesiac unlearning*, is a more focused and precise method that is suitable to removing even a single instance. I apply these methods to address privacy and regulation concerns that arise from data protection regulations such as the *right to be forgotten*.

Unlearning Emperical Evaluation: I present a novel testing scenario to measure the amount of information a model contains about a specific datum using inference measures, model inversion attacks, and membership inference attacks. I use this testing scenario to evaluate the efficacy of the unlearning methods in terms of both model performance and information removal.

Proxy Feature Detection: In the realm of individual fairness, I present a novel detection method for finding *proxy features*, that is, features that convey a significant amount of information about a protected feature. Along with the novel detection method, I propose multiple methods for ranking features in terms of fairness violations and model importance, allowing the user a great amount of control over feature removal.

Automated Individual Fairness Repair: I present an automated repair method for individual fairness that detects fairness violations, diagnoses the responsible proxy variables causing those violations, and repairs the model to effectively increase model fairness without the need to train a new fair model.

The second chapter of this thesis covers Amnesiac Machine Learning, a work which broadly deals with removing learned information about a specific datum or subset of data from a neural network, a task commonly referred to as *unlearning*. This task is applied for the purpose of protecting against attack algorithms that can leak private information, and in doing so protecting user privacy. This work has broad applicability for privacy-sensitive scenarios as well as significant relevance for model holders concerned with compliance toward data privacy regulations as the European Union's Data Protection Regulation. In this chapter I present two unlearning algorithms as well as a novel evaluation framework, and empirically show that both algorithms are effective at removing specific learning from a trained model and protecting against data privacy attacks.

The third chapter focuses on fairness in machine learning. Models can exhibit unfair behavior with regard to a certain protected feature such as gender or race, and this unfair behavior is a significant problem for financial modeling, legal prediction, risk classification, or any other domain where machine learning is responsible for life-influencing decisions such as granting a mortgage or parole. To complicate the issue, this unfairness can exist even if the protected feature is not present in the data through the existence of *proxy features* that contain information about the protected feature. In this chapter I present an end-to-end algorithm that detects individual fairness violations with regard to a certain protected feature, identifies the features most responsible for these violations, and repairs the network to improve fairness while maintaining model performance as much as possible.

The common thread tying these research directions together is that while DNNs are opaque and indecipherable, we are still able to effectively develop methods that sculpt them to our needs. The Amnesiac Machine Learning framework causes neural networks to forget information about certain subsets of training data, despite being unable to determine exactly how that information is learned and retained by the network. In the second chapter, although we cannot determine exactly what parts of training data or the learning process are responsible for unfair behavior, we can still effectively modify that behavior without needing to modify the data or the learning process. In both cases the repair method is model agnostic and requires less training time than simply training a new model from scratch to be suitable for our needs. These methods both give a great amount of control over the repair methods, and enable model holders to feel confident that their models can be modified according to their needs.

1.2 Thesis Organization

This thesis is broadly organized into two primary chapters, and each chapter has it's own methodology, problem statement, evaluation, and conclusion.

The second chapter dealing with *amnesiac machine learning* begins with a description of why the problem of machine unlearning is important, motivated by data protection regulations such as the *right to be forgotten.* I define a problem statement and then present methodology for two novel unlearning mechanisms for deep learning, as well as a naive mechanism that serves as a baseline for comparison. The goal of these methods is to cause the model to behave similarly to one that has never learned on the sensitive data. A properly generalized model should make correct predictions with some high probability on data it has not learned from, and the goal of our unlearning is to end up in a similar state. I define a threat model that serves to outline the type of attacker that could threaten user privacy, and use this threat model as a basis for a novel evaluation scenario that utilizes inference measures, model inversion attacks, and membership inference attacks to estimate the information about some subset of data that remains in a model. I use this evaluation scenario to show that both unlearning mechanisms greatly outperform the naive scenario, and further show the effect of changing levels of information removal on the model performance. I then discuss the related work existing in the field, and position my work in that greater field.

The third chapter deals with the concept of individual fairness. I begin by motivating why fairness is something we should be concerned about through examples of historically unfair prediction making, and broadly describe the notions of fairness that have governed research into the field. I then present an automated repair cycle that, given a trained machine learning model, increases the fairness of that model. This repair cycle does so by highlighting individual fairness violating examples, diagnosing the features most responsible for this unfairness (proxy features), and performs a repair to remove the effect of these features. The user is given the option of three different proxy feature ranking methods, to allow them to balance fairness needs against performance needs according to the fairness / performance tradeoff in machine learning. I then perform an evaluation on a real-world dataset, and show that this repair method greatly increases model performance, producing models that are similar to *fair models* that have never been trained using the identified proxy features. This repair method also takes significantly less time than training a fair model from scratch, showing that it is effective and efficient at producing models with less individual fairness violations.

In the conclusion, I detail the importance of these methods and position them within the greater paradigm of model repair. I take a brief look into what future research could hold, and how the field of machine learning for both research and industry could benefit by advances in this area.

Chapter 2

Amnesiac Machine Learning

2.1 Introduction

In 2016 the European Union (EU) established the *General Data Protect Regulation* (GDPR) which is intended to provide individuals in EU nations control over their personal data. This includes regulations for businesses that handle personal data, requiring them to provide safeguards to protect data and use the highest possible privacy settings by default [6]. In particular, article 17 of the GDPR gives individuals the right to be forgotten and states that "... (businesses) have the obligation to erase personal data without undue delay" [6]. Individuals who invoke this right must have their personal data removed from data records that companies, governments, or other entities hold.

In light of the GDPR law, the existence of attacks on neural networks that leak information about the data they were trained on, such as the Model Inversion Attack [13] and the Membership Inference Attack [39], present a problem for companies or researchers that use personal data to train neural network models. If an individual's data has been used to train a model and that individual subsequently invokes the right to be forgotten, simply deleting the training data is insufficient to protect the individual's privacy. The reason is that neural networks, whose training set contains a specific datum, are vulnerable to leakage of that datum via the above-mentioned attacks. Although this problem is recent, legal scholars have underlined that the potential cost due to GDPR can be significant and potentially crippling to businesses and entities [21],[43]. If a malicious entity or attacker can attack a trained model and learn private information about an individual who has invoked the right to be forgotten, the model owner can be held liable since he or she has not taken the appropriate steps to protect the individual.

In this chapter, I address the question of efficiently removing learned data from a trained neural network without unduly harming the performance of the network. I call this approach amnesiac unlearning and provide extensive empirical analysis to highlight its strengths as well as understand its weaknesses. The obvious solution to this problem is to entirely discard the trained model and train a new one from scratch. Unfortunately, training machine learning models is known to be an expensive and time-consuming proposition. Model owners are likely to be disinclined to continually train new models to be compliant with regulations every time an individual invokes their right to be forgotten. Hence, it is in their best interest to have efficient deletion methods to remove learned data that ensure compliance with regulations and retain the fidelity of their model. Humerick [21] highlights the importance of having a method that effectively removes learned data, as well as the potential economic harm that could be caused by requiring model owners to continually replace trained models.

This work is situated in the broader context of unlearning for machine learning models. While some machine learning techniques such as linear regression can be modified in a mathematically strict way to behave as if they had never been trained on some certain datum, deep learning models that learn in a stochastic and opaque way are not as easy to decipher. Current approaches tend to focus on ways to make retraining easier [2], or to limit the effect that each datum can have on the learning process [10, 9, 4, 1]. By contrast, I instead focus this unlearning approach in the space of practical, post-training methods that do not require specific architecture or data processing choices. Although this comes at the cost of provable data removal, the methods are both efficient and effective. The context this work centres is the domain of privacy, but this approach can be beneficial for any sort of application where removing learning from some specific set of data is required, such as removing data poisoning attacks or undoing the effect of mislabeled training points.

Problem Statement:

This chapter focuses on the following problem: *how can learned data be safely and efficiently removed from a trained neural network without unduly harming the performance of the network?*

Contributions

In this chapter, I make the following contributions:

- I address the problem of how to make neural networks compliant with the right to be forgotten policy of the GDPR law via effective ways of removing learned data. In this context, I introduce two methods for efficient removal of learned data, namely

Unlearning and *Amnesiac Unlearning*. Unlearning is a training-time method that uses relatively little time to remove learned data from a network. By contrast, amnesiac unlearning is a single-step method that is effective for laser-focused removal of the learning from smaller subsets of data (as small as a single example). I compare these methods against the naive method of retraining a model without the sensitive data. I show that these methods are not only very efficient, but effective in removing traces of the data that could be leaked through state-of-the-art attacks, and further do not unduly harm the performance of the neural network.

- I provide a detailed empirical evaluation of the efficacy of my methods along several vectors, including protection against data leaks, efficiency, and model performance. Specifically, I evaluate protection of the proposed methods using state-of-the-art privacy leaking attacks, such as model inversion and membership inference, and show that amnesiac unlearning is the most effective when compared to other methods. As part of this evaluation, I present a modified version of the Model Inversion Attack [13] that is effective even against complex convolutional networks, a class of network previously considered impervious to such attacks [19]. Further, I show that both unlearning and amnesiac unlearning are very efficient to apply and cost very little to model owners. Finally, I evaluate the performance of neural network models after the application of these deletion methods, and show that these methods have virtually no impact on model performance on data that is unrelated to the deleted data.

2.2 Methodology

In this section, I cover the methods for removing learned data from a neural network model. I begin by presenting the basic scenario of simply removing sensitive data from the training set and training the model on this new dataset, and use this naive approach as a baseline of comparison for the *unlearning* and *amnesiac unlearning* methods that are both more efficient and more effective at removing learning. I then present a general threat model for attempting to leak private information from a model, and discuss how it can be used to evaluate the methods.

2.2.1 Naive Retraining

As a baseline, I consider the naive method of simply removing the sensitive data from the dataset and continuing model training without the sensitive data. The principle of catastrophic forgetting [36] tells us that when presented with new data, there is some nonzero probability of a neural network losing previously learned information. A network originally trained on a dataset D and subsequently trained on $D \setminus S$ has a high probability of losing information about S. Although this has a high probability of causing the neural network to forget the desired information, it gives no information or assurances about how long it could take. However, it provides a useful baseline because simply removing the data and retraining the network is a simple solution and, if useful, provides model owners with an easy method to cause networks to forget sensitive data. Our evaluation shows that naive retraining takes far too long to be considered as a practical solution and even a large amount of naive retraining does not prevent data leaks.

2.2.2 Unlearning

The goal of unlearning is to muddy the model's understanding of the sensitive data to the point that it is unable to retain any meaningful information about that data. To achieve this, we relabel the sensitive data with randomly selected incorrect labels and then continue training the network for some iterations on the modified dataset. Unlearning for an entire class is achieved by replacing the label for each example in that class with a randomly selected incorrect label, whereas unlearning for a select set of examples is achieved by removing those examples and inserting a small number of copies of each of them with randomly selected incorrect labels. This relabeling is computationally inexpensive and our evaluation shows that this method is effective with only a very small number of training iterations on the modified dataset.

One possible risk of unlearning is that the data holder must maintain a copy of the sensitive data during the unlearning process, which may potentially have legal significance. The right to be forgotten currently gives data holders up to a month to remove the data [22], which makes this method easily fall within permissible bounds, but other regulatory bodies may impose standards that limit the effectiveness of this method. Additionally, the relabeling and retraining leads to the model displaying unintuitive behavior at those data points, making this a potential new attack vector.

2.2.3 Amnesiac Unlearning

Amnesiac unlearning involves selectively undoing the learning steps that involved the sensitive data. During training, parameter update logs are kept that record parameter changes as well as the indices of included examples. When a data removal request comes, the model owner undoes the parameter updates from only the batches containing sensitive data. As long as the number of batches effected is small, the effect of undoing those sections of learning is also small. One significant advantage of this method is that it can very effectively remove the learning from a single example with minimal impact on the rest of the learned model, which is something other techniques have struggled with. Further, this method is very time efficient, and our evaluation shows that as long as the amount of data to remove is small the efficacy of the model remains unharmed.

To better understand amnesiac unlearning, we can revisit model training with a fresh perspective. We can view model training as a series of parameter updates to the initial model parameters (that, in the case of neural networks, are randomly initialized). We begin from initial model parameters $\theta_{initial}$. Model M is then trained for E epochs each consisting of B batches, and the parameters are updated after each batch by an amount $\Delta_{\theta_{e,b}}$. The learned model parameters can then be expressed as:

$$\theta_M = \theta_{initial} + \sum_{e=1}^{E} \sum_{b=1}^{B} \Delta_{\theta_{e,b}}$$

During training, we keep a log SB of which batches contained the sensitive data. This can be in the form of an index of batches for each example in the training data, an index of batches for each class, or any other form desired. We also must maintain the model parameter updates from each batch that contained sensitive data. A comprehensive approach would be to keep a record of each batch update. However, if the data holder is only concerned about possible potential removal of a subset of data, they need only keep the parameter updates from batches containing that data.

Once training is completed, a protected model M' can be produced using amnesiac unlearning as follows: subtract the parameter updates from each batch $sb \in SB$ (the list of sensitive data batches) from the learned parameters θ_M.

$$\theta_{M'} = \theta_{initial} + \sum_{e=1}^{E} \sum_{b=1}^{B} \Delta_{\theta_{e,b}} - \sum_{sb=1}^{SB} \Delta_{\theta_{sb}} = \theta_M - \sum_{sb=1}^{SB} \Delta_{\theta_{sb}}$$

Note that the parameter update at each step $\Delta_{\theta_{e,b}}$ depends on the current parameters when that learning step is taken. Due to this, a model trained on three batches b_0, b_1, and b_2 that then has the parameter updates from b_1 undone will not in fact be the same as a model (with the same initial parameters) trained directly on b_0 and b_2, because the parameter updates from learning on b_2 will be different.

When the number of batches in SB is low, the difference is between θ_M and $\theta_{M'}$ is comparatively low, and the concomitant impact on the efficacy of the model is also low. In this way, this method provides a way of laser-focused removal of sensitive data, i.e., the learned data from a single record can be removed from the model with minimal effect on the rest of the model. This method is particularly well-suited for situations where the privacy of an individual with a single record in the dataset needs to be protected. When the number of batches in SB is higher, the model needs a small amount of fine-tuning after the amnesiac unlearning step to regain performance.

One potential downside to this method is the large storage space required to keep a set of parameter update values from each batch. While this cost can be quite large, especially for state-of-the-art large models, the low cost of mass storage means this cost is usually less than the cost of retraining a full model from scratch. Model owners concerned about this storage cost may be better off using a method such as unlearning that doesn't require this storage overhead.

2.2.4 Threat Model

We use two state-of-the-art attack methods to evaluate how much data can be leaked, namely, model inversion and membership inference attacks. The adversary attempts to gain information about either the class (through model inversion attacks) or about the presence of a specific record or set of records in the training data (through membership inference attacks). Leaking class information could provide an attacker with generalized information they should not have. For example, if the adversary learns that the class of individuals that are at risk for cancer is strongly generalized to be older men, it could potentially violate privacy guidelines that protect the older men represented in the data set. Leaking membership information could also pose a significant privacy threat. For example, if an adversary learns that an individual was represented in a dataset used to train a model to determine bankruptcy risk, they could determine the individual's private bankruptcy history.

We consider the scenario where the adversary has white-box access to the currently published version of the model, but does not have access to any previously published versions. In the case of the model inversion attack the adversary does not have information about what each class represents, and we consider an attack successful if the adversary is able to glean information about what the class represents through model inversion. In the case of the membership inference attack the adversary has access to data from a similar distribution to the one used to train the target model. The data used to train the target model and the data available to the adversary can contain duplicate records, but does not have to.

2.3 Empirical Evaluation

I have conducted extensive experiments to evaluate the efficacy of our unlearning methods. First, I present an evaluation of the model accuracy during our methods, looking at both the model accuracy of the target data (that is marked for removal) as well as the accuracy on the non-target data (sometimes referred to as model performance). Second, I present the results of model inversion attacks against the models at different stages of the data removal process to show how data can be retained and subsequently leaked for a considerable time period after model accuracy starts to degrade and to evaluate the efficacy of the data removal methods to stop this data leaking. Next, I present the results of membership inference attacks against models before and after data removal to show the effectiveness of data removal methods against record-level data leaking. Finally, I show the effect of different levels of amnesiac unlearning on model accuracy, showing the relationship between amount of amnesiac unlearning and model degradation.

Experimental Environment: All algorithms are implemented in Python 3.7 and use the PyTorch deep learning library [32]. All experiments were conducted on the Amazon Sagemaker platform using an ml.g4dn.xlarge instance with 4 vCPUs, 1 GPU, and 16GB of memory.

Datasets: Experiments were conducted on the following two well-known datasets. These datasets were chosen because of the ubiquity of experiments using them as well as to highlight the performance of our algorithm against tasks of varying complexity.

1. **MNIST handwritten image dataset** [26] is a widely used 10-class dataset consisting of 60,000 training images and 10,000 testing images. These images are in grayscale with a resolution of 1x28x28 pixels each.

2. **CIFAR100** [24] is a 100-class dataset consisting of 600 images from each class. Classes are varied and consist of objects and living things such as dolphins, sunflowers,

bottles, and trains. The images have 3 colour channels and have a resolution of 3x32x32 pixels each.

Neural Network: All experiments were performed on the Resnet18 convolutional neural network [18], a state-of-the-art residual learning architecture.

Attack Algorithms: Here I describe the two state-of-the-art attack methods considered in this chapter.

1. **Model Inversion Attack:** The model inversion attack used for these evaluations is a modified version of the standard model inversion attack seen in Fredrikson et al. [13]. The standard attack is deterministic, beginning with a feature vector with all features assigned to 0 (or a suitable starting point for the domain) and labeling this feature vector with the label of the target class y_t. The gradient of loss with regard to the feature vector is then calculated, and the value of the gradient multiplied by a learning rate is subtracted from each feature. This process is repeated iteratively, altering the feature vector to be more similar to what the model considers to be an example from the class y_t.

In the original model inversion attack, a *PROCESS* function is performed after each gradient descent step and is intended to help recognition by performing some image processing. I found that this detracted from clarity of generated images, and instead the image processing is periodically applied every k gradient descent steps (where $k \in [500, 1000]$). The original attack also begins each inversion from the same state, making each attack deterministic. However, the results are more varied when starting each inversion with a small amount of noise added to each feature, so each inversion is different. Finally, the original attack continues until the change in loss is below some threshold, while the attack is often more effective if continued the attack for some set number of iterations, even while the change in loss is small. These modifications allowed for generated inversions even on complex convolutional architectures such as Resnet18, a task that was previously judged to be infeasible [19]. The model inversion attack is presented in Algorithm 1.

Algorithm 1 Model Inversion Attack

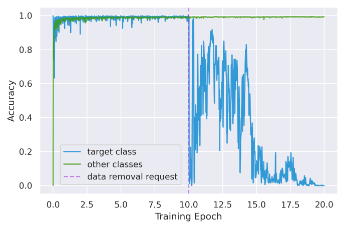
```
1: procedure INVERSION($f, target$)
2:     $x_0 \leftarrow \mathcal{N}(\mu, \sigma^2)$
3:     for $i \in [0, n)$ do                                    ▷ $n$ iterations
4:         $L \leftarrow Loss(target, f(x_i))$
5:         $x_{i+1} \leftarrow x_i - \alpha \cdot \Delta_x L$              ▷ Update the example w.r.t. loss
6:         if $i \% k == 0$ then  $x_{i+1} \leftarrow PROCESS(x_{i+1})$
7:     return $x_n$                                        ▷ The inverted example
```

2. **Membership Inference Attack:** I implemented the membership inference attack as described in Yeom at al. [45]. As described, the attack dataset was created by taking the softmax prediction vector $\mathbf{y} = f_{shadow\_model_i}(x)$ from each example x and creating a dataset D_y for each class y, where each example is a prediction vector for an example in that class $\mathbf{y}$ and each label is either a 0 or 1, depending on if the example was in the training data for that shadow model. An attack model was then trained on each dataset, and this suite of attack models was used for testing. The attack model is a fully connected network with two hidden layers of width 256 and 128 with ReLU activation functions and a sigmoid output layer. Increasing the number of shadow models increases the accuracy of the attack and the computational cost.

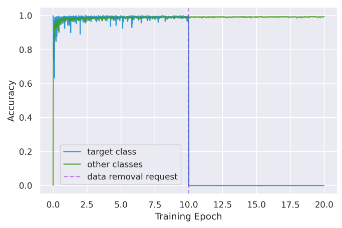
2.3.1 Model Accuracy on Target and Non-target Data

MNIST: Figure 2.1a shows the model performance for naive retraining. The model retains a significant amount of knowledge about the sensitive target data for a long period, and the model accuracy on the target data drops slowly over the retraining period. Conversely, figure 2.1b and figure 2.1c show the model accuracy for the sensitive target data drops extremely rapidly for both the unlearning and amnesiac unlearning methods. In the amnesiac unlearning method, the model accuracy on the non-target data takes a slight dip when the batch learning is initially reversed, but a small amount of training corrects that.

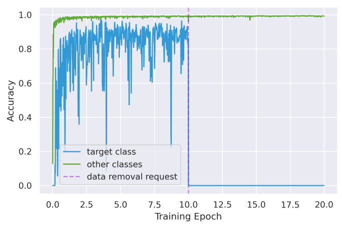
CIFAR100: As seen in figure 2.2a, the naive retraining method here is even less effective than in the MNIST setting, and after 10 retraining epochs the model still has an approximately 40% prediction accuracy on the target data. By contrast, the unlearning method (shown in figure 2.2b) shows a much steeper drop in accuracy on the target data, and within 2 epochs the model has a very small prediction accuracy. After 5 epochs, the unlearning method removes virtually all ability for the model to correctly recognize the target data. The amnesiac unlearning method (shown in figure 2.2c) immediately removes the ability for the model to recognize the target data. In both the naive and unlearning methods the prediction accuracy of the non-target data stays high, showing that these methods do not degrade the learned information for non-target data. The amnesiac unlearning method shows an initial accuracy loss when the batches are reversed, but this quickly rebounds after a small amount of training. In this instance, approximately 6% of the batches were removed, which makes for a significant effect on the rest of the model. This effect is examined in greater detail in the **Effect of Amnesiac Unlearning on Model Efficacy** section.



(a) Naive Retraining

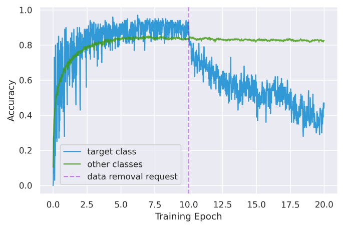


(b) Unlearning

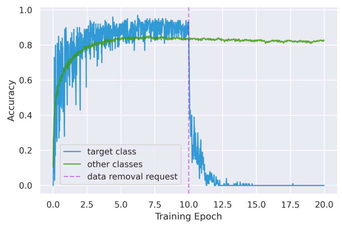


(c) Amnesiac Unlearning

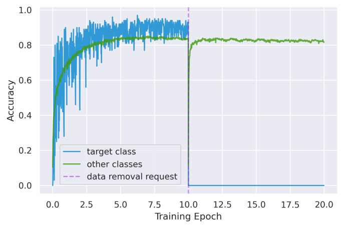
Figure 2.1: MNIST model accuracy on target and non-target data



(a) Naive Retraining



(b) Unlearning



(c) Amnesiac Unlearning

Figure 2.2: CIFAR100 model accuracy on target and non-target data

2.3.2 Model Inversion Attacks

Model inversion attacks were performed on trained MNIST models both before and after the data removal methods were applied. Each model started from the same before state, and an inversion attack against the trained model can be seen in figure 2.3. Further inversion attacks were performed during and after the data removal process. In the case of the naive retraining and unlearning methods, inversion attacks were performed after each epoch of training on the dataset modified to cause data removal. In the case of amnesiac unlearning, the amnesiac unlearning method was performed and then inversion attacks were performed after each epoch of training on a modified dataset without the target class. All inversion were performed targeting the class representing the digit **3**, and all data removal methods likewise attempted to remove knowledge of that class. Due to the stochastic nature of our modified inversion attack, we performed multiple attacks against each model and selected the ones that most resemble the target class. Unfortunately, it is very difficult to come up with an acceptable metric to quantify the ”recognizability” of inverted images, and visual recognition was our best way of judging the success of the attacks.

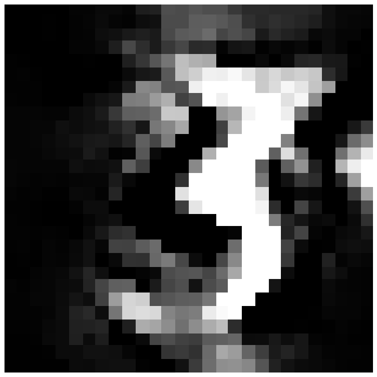


Figure 2.3: Model Inversion Attack on trained model

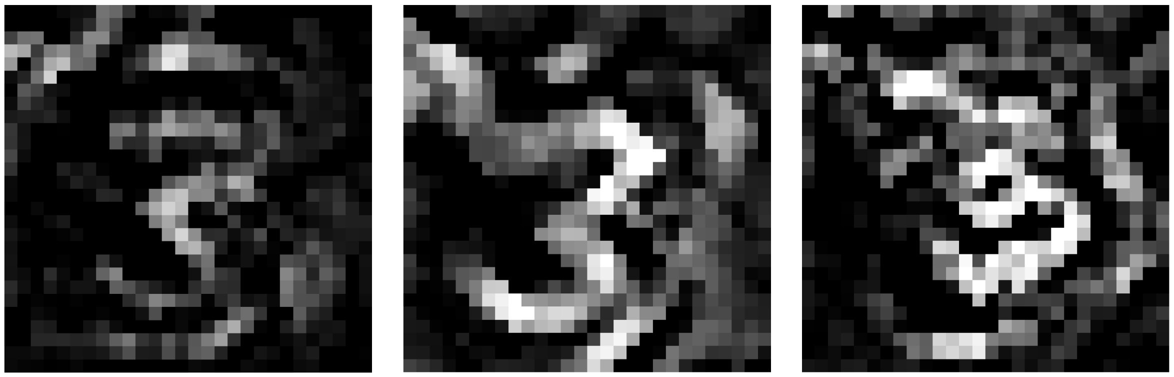


Figure 2.4: Model Inversion Attack results after 1, 5, and 10 epochs of naive retraining

As seen in figure 2.4, the naive retraining method does very little to protect against leaking private class information. Even after 10 retraining epochs, inversions targeting the class are still somewhat recognizable. In conjunction with the test accuracy results, this underlines the unsuitability of naive retraining to protect private information.

The model inversion attacks results against the models effected by unlearning can be seen in figure 2.5. This method almost immediately completely removes any ability to gain useful class information with model inversion attacks, showing how effective this method is at protecting against this sort of data leaking attack.

For the amnesiac unlearning method, the model inversion attacks have remarkably little gradient information to go off of, and as a result the images (seen in figure 2.6) are dark and jumbled, although they do seem to hint toward the general shape of the target data. As

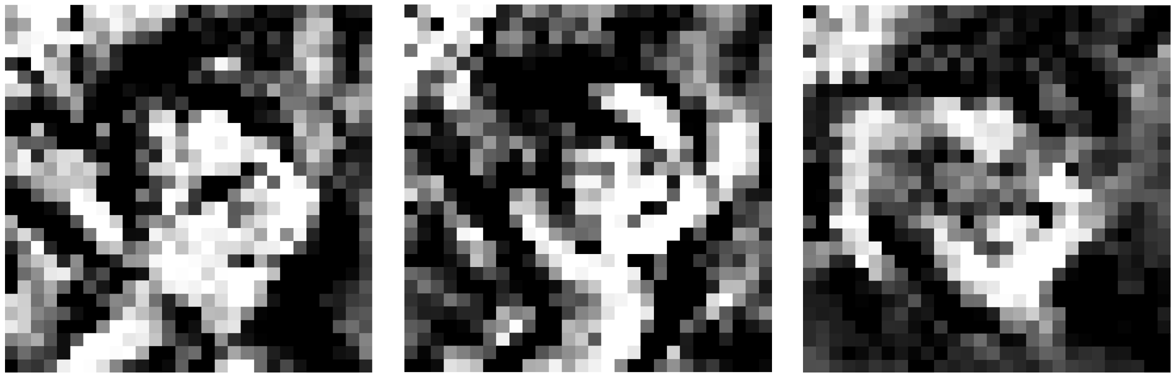


Figure 2.5: Model Inversion Attack results after 1, 5, and 10 epochs of unlearning

expected, more training on the modified dataset removes the ability to gain any meaningful information about the target data, and the attacks after 5 and 10 epochs of training are almost unrecognizable. This suggests that this method is best used in situations where a small number of data points needs to be removed, as opposed to an entire class.

2.3.3 Membership Inference Attacks

Membership inference attacks were performed with 16 shadow models, each trained on CIFAR100 for 10 epochs. The target model was likewise trained for 10 epochs. We evaluate the effectiveness of membership inference attacks using the recall metric, a metric that give us great insight into how effective these attacks are at leaking data. This metric is more helpful than accuracy in an environment where we care a lot about correctly recognizing positive instances. This gives us information about how effective our method is at preventing membership inference data leaks. All membership inference attacks were performed targeting a set of individual examples, and then the data removal techniques were performed to attempt to remove learned data from this set of individual examples.

Membership inference attacks were performed on a trained model, and the recall value of this attack can be seen in table 2.1 at epoch 0. Data removal techniques were then performed, and the result of membership inference attacks on the model after the application of the removal technique but before any retraining can be seen at epoch 0'. In the case of naive retraining and unlearning, this is simply changing the datasets and has not had an effect, while in the case of amnesiac unlearning it represents the reversal of the batches

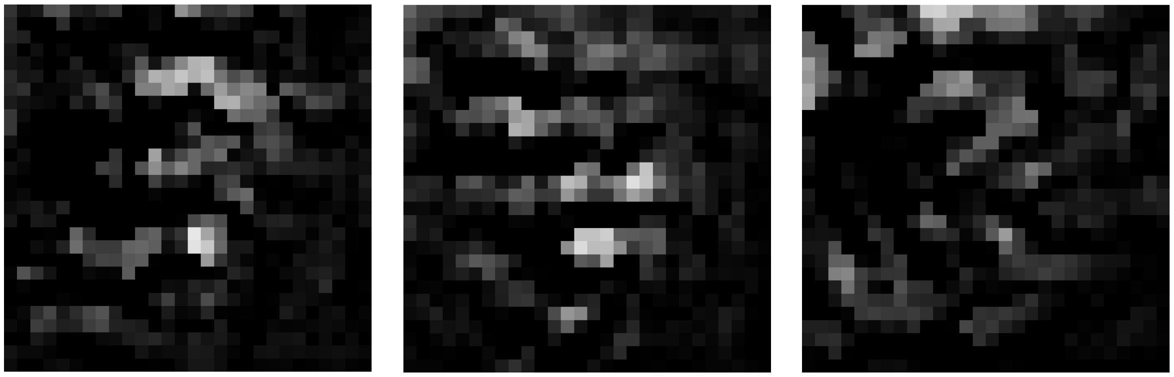


Figure 2.6: Model Inversion Attack results after 1, 5, and 10 epochs after amnesiac unlearning

Epoch	Naive Retraining	Unlearning	Amnesiac Unlearning
0	0.97478991	0.97478991	0.97478991
0'	0.97478991	0.97478991	0.0
1	0.09243697	0.0	0.0
2	0.05882352	0.0	0.0
3	0.0	0.0	0.0
4	0.0	0.0	0.0
5	0.0	0.0	0.0

Table 2.1: Recall of membership inference attacks

containing the sensitive data. Subsequent membership inference attacks were performed after each epoch of training on the modified datasets.

The results shown in table 2.1 show that the naive retraining method does not prevent the membership inference attacks for more than 2 full epochs of retraining. This highlights the insufficiency of this method to protect against data leaking attacks, and emphasizes the need for other methods. By contrast, the unlearning and amnesiac unlearning methods both protect against membership inference attacks with less than an epoch of retraining (and in the case of amnesiac unlearning, with no retraining whatsoever). In conjunction with the performance of these methods against the membership inference attack, it is clear that both of these methods are effective against these state-of-the-art privacy leaking attacks.

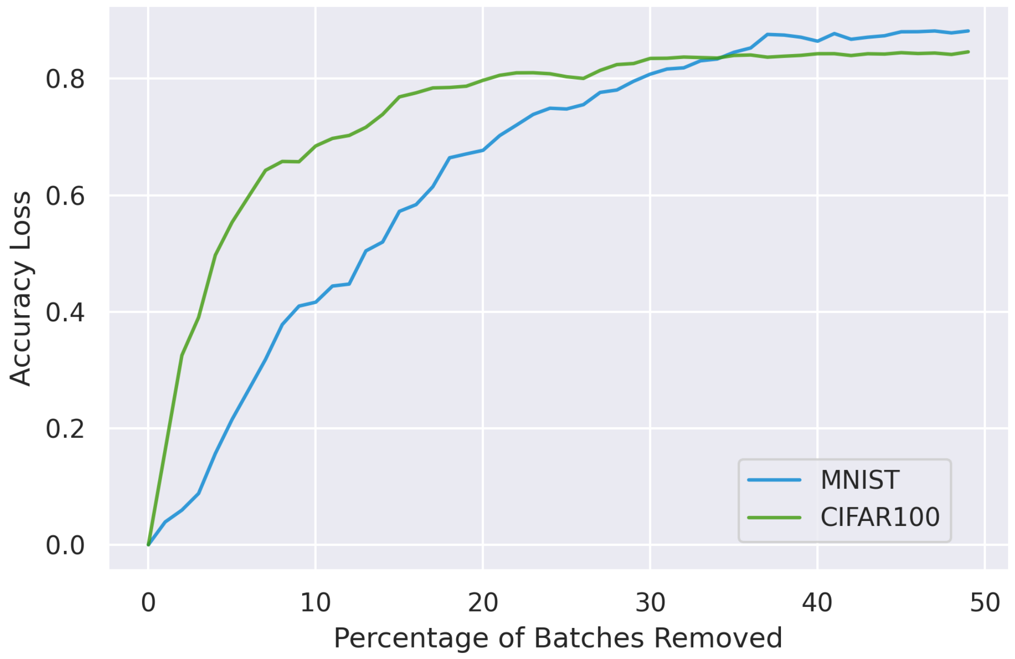


Figure 2.7: Test accuracy on all classes with increased amnesiac unlearning (mean of 10 runs)

2.3.4 Effect of Amnesiac Unlearning on Model Efficacy

Amnesiac unlearning, while extremely effective at removing the learning from specific data examples, has a significant effect on the efficacy of the model if it is used too often. With 1% or less of the batches removed the model accuracy remains close to what it was before any amnesiac unlearning was performed, but this quickly drops as the learning from more batches is removed, as seen in figure 2.7. Drops in accuracy can be easily remedied with a small amount of fine-tuning, but this underscores the need for the user to be wise in selecting their data removal method - if they have a small amount of data that needs to be removed, amnesiac unlearning is efficient and effective, without significantly effecting the rest of the model. However, if a larger amount of data needs to be removed, consideration should be given to unlearning, which is able to remove larger amounts of learned data without needing the same fine-tuning after the removal step.

2.4 Discussion

Metrics: We emphasize that while test accuracy, model inversion attacks, and membership inference attacks can give useful information about a model's propensity to leak private information about sensitive data, they are not methods of measuring how much information has been retained about that sensitive data. Some models and settings aren't vulnerable to model inversion attacks and membership inference attacks can be protected against with methods such as differential privacy, but neither of these things provide a guarantee that models cannot leak other private information. However, finding a comprehensive method of evaluating how much private data can be leaked from a model through any means is a difficult task, and to-date we are not aware of any method that claims to do so.

Black-box Attacks: One may argue that the impact of model inversion and membership inference attacks can be mitigated by allowing only black-box access to the models themselves. Unfortunately, model extraction attacks [23], [41], [38] have the ability to steal functionality of models even with only black-box access, and in some cases even with only access to truncated prediction vectors [30]. The existence and success of these attacks show that limiting access to black-box access alone is not sufficient to protect against motivated attackers who can steal a model and then attack it under a white-box setting.

2.5 Related Work

Machine learning models have been shown to leak information about the data they've been trained on [20, 39, 14, 13, 19]. Specifically, two main kinds of information leaking attacks have been studied: *membership inference attacks* that leak information about the presence of specific records in the training data [45] and *model inversion attacks* that leak class information [13],[19]. Membership inference attacks determine whether a particular record was present in the training data for a model. This attack was first presented in 2008 [20] and was formalized in 2015 [11]. Since then, considerable work has been done on membership inference attacks and defense mechanisms against such membership inferences [28],[39],[45]. Property inference attacks are a subset of membership inference attacks that determine a general property of the training data, such as the ratio of training examples in each class [15]. Model inversion attacks, introduced by Fredrikson et al. in 2014 [14] and expanded to vision tasks in 2015 [13], have been shown to recreate instances of records from trained ML models. Given white-box access to a trained model, examples of target classes or points near a regression value can be recreated. In this work, I utilize both state-of-the-art membership inference attacks and model inversion attacks to evaluate how likely a model is to leak private data.

There is an abundance of literature on differential privacy which provides an upper bound on the amount of information that can be learned from each individual data record [10],[9],[4],[1]. Cummings and Desai address the need for training machine learning models in a differentially private manner to comply with GDPR [7]. However, differential privacy methods do not allow learned data to be forgotten, they simply provide assurances that each individual data record does not contribute enough to be leaked with confidence. Moreover, to implement differential privacy with a reasonable bound on potential privacy risk often requires significantly reduced predictive ability. Any of the methods presented in this chapter can be combined with differential privacy methods if this type of privacy guarantee is needed.

Other recent research has touched on the issue of removing learned properties or data from machine learning models. Ginart et al. [16] devised a notion they term *removal efficiency* and give two algorithms for efficiently removing specific data points from k-means clustering models. Guo et al. [17] defined an approach they term *certified removal* that was evaluated for linear classification models. In this system, a model is trained on a dataset including sensitive data, and then the sensitive data is removed in some way from the model. A different model is trained on the same dataset without the sensitive data, and removal algorithms are evaluated based on the difference between the models. However, these approaches are unlikely to work in the case of DNNs, which are more opaque and difficult to analyze. The work presented in Ginart et al. [16] and Guo et al. [17] are both focused on such different machine learning methods that it is impossible to make a direct comparison with this work.

Recently, Bourtoule et al. [2] introduced a method for dealing with individual data removal requests. They proposed SISA training, a method consisting of an aggregate model made of multiple models trained on disjoint partitions of the data. Because of this segmentation, when requests for removal are made the model owner can retrain only the effected sub-models instead of having to retrain everything. By contrast, these methods focus on deep learning models that have already been trained and are independent of neural network architecture. Further, this method does not require that the model is an ensemble model using an aggregate of weak learners.

2.6 Conclusion

In this chapter, I examined and evaluated methods aimed at removing learned data from trained neural network models. This is especially relevant because of right to be forgotten regulations, such as in the European Union's GDPR law, that require data holders to delete data on individuals when requested. Training new models from scratch can be prohibitive, and there is an acute need for methods that can effectively and efficiently remove learned data from the models to protect user privacy, while at the same time preserving model performance on non-target data.

Considering the problem of finding a way to repair a model so it is more suitable for a certain goal without harming a model's performance, this is a very clear success at improving model privacy over a subset of data while maintaining the efficacy of the original un-private model. This represents a solid move forward in finding solutions for the problem, although this is only one small aspect of the larger line of research of model repair.

To solve this aspect of the problem, I introduce two novel methods of data removal, namely *unlearning* and *amnesiac unlearning*, that can be used to protect privacy of target (sensitive) data without incurring significant cost or degrading model performance on non-target data. I evaluated these methods against two state-of-the-art data leaking attacks, namely model inversion and membership inference, and showed that both data removal methods are effective at protecting data from leaking. An additional interesting finding was that the unlearning method is better for removing large amounts of learned data, while amnesiac unlearning is better for laser-focused removal of specific sections of data, such as a single example or set of examples. Given that this line of research at the intersection of data privacy and machine learning is a relatively new and very rich field, there remains a number of important problems to be solved, including a method of measuring data retention after data removal techniques are applied.

Chapter 3

Automated Model Repair for Individual Fairness

3.1 Introduction

Machine learning models are continually taking a larger role in controlling decisions that impact our lives over a range of areas from mortgage approvals [44] to sentencing decisions[35]. Despite this ubiquity, models can display significant bias or unfairness when making decisions [29, 35, 46]. This bias is often present in model predictions even when the protected characteristic is not included in the feature set. The source of this bias can be difficult to detect due to the existence of *proxy features* that convey information about protected features. The concept of other features acting as proxies for the protected characteristic is so common it is sometimes referred to as digital redlining [37], referencing historical examples of financial institutions that would mark black neighborhoods with red lines and deny mortgages to that region after denying based on race was outlawed. When model predictions have such a significant impact on decisions that influence people's lives, careers, and futures, there is a great moral imperative to ensure that their decisions are not deepening existing inequalities or treating disadvantaged demographics unfairly.

Generally, we view fairness in terms of either *group fairness* metrics that deal with treatment of entire demographics, or as *individual fairness* metrics that govern how individuals should be treated with regard to membership in a protected demographic. Individual fairness is often more difficult to detect, due to the influence of proxy features or other difficult to discern sources of bias.

In this work, I present an end-to-end model repair method for detecting individual fairness violations, discerning which features are acting as proxy features contributing to this unfairness, and repairing the model to mitigate unfairness in inference. This repair method is intended for deep learning and allows for flexibility in terms of model architecture and removal priority, allowing users the choice of how to prioritize fairness and predictive performance. I evaluate this repair method against both the unrepaired models and a model trained from scratch without the identified proxy features, and show this repair method generates models that are similar in fairness and performance to the model trained from scratch, yet requires significantly less resources.

Contributions

In this chapter, I make the following contributions:

- A novel method for detecting proxy features using an *auxiliary model* trained to predict a protected feature, along with multiple ranking methods to prioritize proxy feature removal according to needs
- A model repair method that removes the influence of identified features without the need to train a new model
- An automated model repair method for individual fairness that automatically detects proxy features and removes the influence of them while maintaining model performance, improving individual fairness at significantly less computational cost than training a model from scratch

3.2 Notions of Fairness

Much research on bias and fairness with regard to machine learning divide ideas into two distinct notions of *group fairness* that measures treatment toward a certain demographic as a whole, and *individual fairness* measures that govern how an individual should be treated with regard to membership in a given demographic. These notions are often at odds with each other; for example, the individual fairness concept of unawareness (that demographic membership should not be a feature) can be incompatible with the group fairness concept of demographic parity (that equal proportions from each demographic should be predicted positively). This distinction carries a significant responsibility for educated decisions about what will be most suitable to ensure demographics are treated fairly. A feature that directly denotes membership in a protected demographic is referred to as a **protected feature**.

Definitions:

- X: A dataset of individual examples
- $x \in X$: A set of features corresponding to an individual example
- Y: A dataset of ground truth labels corresponding to individual examples
- $y \in Y$: The label corresponding to an individual example
- $M(x)$: The prediction of model M on input x
- c: A *protected feature* denoting membership in a protected demographic
- ϵ: An optional threshold value for softening properties

3.2.1 Group Fairness Conceptions

Group fairness conceptions aim to ensure that two demographic groups are treated fairly as a whole. These attempt to give fair treatment to groups that may have been historically disadvantaged or for some reason have a different representation in positive predictions.

Demographic parity ensures that the positive prediction rate for each demographic must be equal. If 80% of individuals from group A are predicted positively, then an equal proportion of individuals from group B must be predicted positively. That is,

Demographic Parity: $P(M(x) = 1|c_x = 0) = P(M(x) = 1|c_x = 1)$

However, this notion can in fact lead to unfair treatment where we may deny some qualified individuals from one group and approve some unqualified individuals from another. Further, it does not consider predictive accuracy: randomly selecting equal proportions from both demographics achieves group fairness while obviously being unsuitable for any predictive task. To help some of these issues, we can instead use the notion of **accuracy parity**. Meeting this criteria ensures that an equal amount of qualified individuals from each group are classified positively, and an equal amount of unqualified individuals from each group are classified negatively. That is:

Accuracy Parity: $P(M(x) = y|c_x = 0) = P(M(x) = y|c_x = 1)$

This can, however, allow us to have similar accuracy values with significantly different confusion matrices, giving a larger proportion of false positives to one group and a larger proportion of false negatives to another, with a result of unequal treatment. We can slightly soften our definition and avoid some of this with the notion of **positive predictive parity**:

Positive Predictive Parity: $P(M(x) = 1|c_x = 0, y_x = 1) = P(M(x) = 1|c_x = 1, y_x = 1)$

Alternately, a stronger alternative to accuracy parity combines positive predictive parity and negative predictive parity:

Predictive Rate Parity: $P(M(x) = 1|c_x = 0, y_x = 1) = P(M(x) = 1|c_x = 1, y_x = 1) \wedge P(M(x) = 0|c_x = 0, y_x = 0) = P(M(x) = 0|c_x = 1, y_x = 0)$

3.2.2 Individual Fairness Conceptions

Individual fairness conceptions are concerned with ensuring individuals that are similar in non-demographic ways are treated similarly. Of these concepts, the most straightforward is **unawareness**, or simply ensuring that the protected feature is not present in the data.

Unawareness: $\forall x \in X, c \notin x$

A different concept which allows for demographic membership as a feature, yet places restrictions on the effect of this feature is is the concept of **individual fairness**, which states that the probability of a prediction must be equal for two individuals that differ only on the protected feature. That is:

Individual Fairness: $\forall x \in X, P(M(x, c) = 1|x, c = 1) = P(M(x, c) = 1|x, c = 0)$

More recently, Kusner et al. [25] developed a notion deemed **counterfactual fairness** that considers the likelihood that causal variables can introduce unfairness with regard to a certain demographic even if that demographic is not at greater risk. As an example, an insurance company that charges higher rates for red cars can unfairly overcharge a racial demographic that disproportionately own red cars, even if that demographic does not have a higher accident rate. The notion of counterfactual fairness considers that, for an individual to be treated fairly, they must have equal predictions when the demographic and demographic-dependent features (x_c) change.

Counterfactual Fairness: $\forall x \in X, \forall x_c, P(M(x, x_c, c) = 1|x, x_c, c = 1) = P(M(x, x_c, c) = 1|x, x_c, c = 0)$

More recently, Castiglione et al. developed an individual fairness notion they refer to as **fAux fairness**. fAux fairness determines that the set of features x was generated through a function f_g acting on two sets of latent variables: z which is independent of the protected feature c and z_c which is dependent on c. Thus, the feature set x is generated through $x = f_g(z, z_c)$. In a similar manner to the notion of counterfactual fairness, their definition of fairness thus depends on equality of prediction with regard to z. That is:

fAux Fairness: $\forall z \in Z, \forall z_{c_i}, z_{c_j}, P(M(f_g(z, z_{c_i})) = 1|z, z_{c_i}) = P(M(f_g(z, z_{c_j})) = 1|z, z_{c_j})$

This method is a variant of individual fairness, and is the basis for the individual fairness detection methods in this work.

Each of these definitions attempt to ensure that individuals are treated fairly with regard to membership in a demographic. However, these methods do not ensure that demographics as a whole are treated fairly. This difference between individual fairness and group fairness effects underscores the responsibility to examine the desired outcome and treatment of demographics and choose the best one for the task at hand. In fairness, as with algorithms, there is no free lunch.

3.2.3 Softened Definitions

Most of the previously defined definitions can be softened, and in practice often are. Softening these allows for some natural noise in predictions and data, and gives a significant amount of control over the outcome through the choice of ϵ value. The softened group fairness definitions tend to give a small amount of difference between predictions over demographics, while the individual fairness metrics allow for a small amount of difference based on the protected feature (and possibly any dependant variables). Softened definitions are as follows:

Softened Demographic Parity: $|P(M(x) = 1|c_x = 0) - P(M(x) = 1|c_x = 1)| < \epsilon$

Softened Accuracy Parity: $|P(M(x) = y|c_x = 0) = P(M(x) - y|c_x = 1)| < \epsilon$

Softened Positive Predictive Parity: $|P(M(x) = 1|c_x = 0, y_x = 1) - P(M(x) = 1|c_x = 1, y_x = 1)| < \epsilon$

Softened Predictive Rate Parity: $|P(M(x) = 1|c_x = 0, y_x = 1) - P(M(x) = 1|c_x = 1, y_x = 1)| < \epsilon \wedge |P(M(x) = 0|c_x = 0, y_x = 0) - P(M(x) = 0|c_x = 1, y_x = 0)| < \epsilon$

Softened Individual Fairness: $\forall x \in X, |P(M(x, c) = 1|x, c = 1) - P(M(x, c) = 1|x, c = 0)| < \epsilon$

Softened Counterfactual Fairness: $\forall x \in X, \forall x_c, |P(M(x, x_c, c) = 1|x, x_c, c = 1) - P(M(x, x_c, c) = 1|x, x_c, c = 0)| < \epsilon$

Softened fAux Fairness: $\forall z \in Z, \forall z_{c_i}, z_{c_j}, |P(M(f_g(z, z_{c_i})) = 1|z, z_{c_i}) - P(M(f_g(z, z_{c_j})) = 1|z, z_{c_j})| < \epsilon$

3.3 Automated Model Repair for Individual Fairness

This chapter presents an approach that is an automated repair method. This method finds points that violate individual fairness using the fAux gradient alignment method, detects the proxy features most responsible for this unfairness, and repairs the model to remove the effect of these features. A user can use this method to increase the fairness of a trained model with regard to a protected feature.

We can view this method as a series of successive steps:

1. Detect individual fairness violating examples
2. Detect proxy features
3. Repair model
4. Validate repair

The end result is a repaired model that is more fair than the original while still maintaining a high level of predictive accuracy. The user maintains a fair amount of control over how proxy features are selected for removal, and as a result is able to prioritize repair to either greatly increase fairness at moderate performance cost, or to increase fairness while also maintaining a highly performing model.

3.3.1 Detecting Fairness Violations

I use the fAux conception of individual fairness as a local detection technique for regions that violate individual fairness. Although the latent representations z (conditionally independent of c) and z_c (conditionally independent on c) are unavailable, we can approximate the local conditional independence around a point x using the notion of *gradient alignment*.

For a model to be fair, conditional probability with regard to the label y and protected feature c should hold when observing an example x (within some threshold). That is, $|P(y,c|x) - P(y|x)P(y|c)| < \epsilon$. We can consequently impose a local independence criterion (LIC) with regard to trained model M: $|\frac{\delta M(x)}{\delta c}|_\infty < \epsilon$. This LIC can be calculated using the chain rule : M: $|\frac{\delta M(x)}{\delta x}\frac{\delta x}{\delta c}|_\infty < \epsilon$. However, since $\frac{\delta x}{\delta c}$ is not calculable without access to the underlying data that generated the features x from protected attributes c, we must approximate this value. This can be done by training an **auxiliary model** M_{aux} that

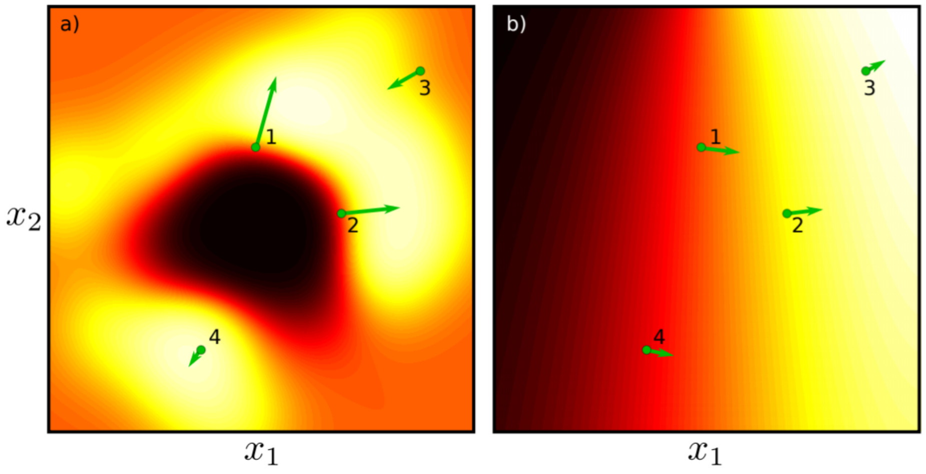


Figure 3.1: Gradients of $M(x)$ (left) and $M_{aux}(x)$ (right) at 4 points.

predicts c given x. The derivative of this model, easily calculable through backpropagation, can provide an efficient approximation for $\frac{\delta x}{\delta c}$. This resulting approximation is referred to as the *gradient alignment* with regard to an example x. For a proof of these steps, please see Castiglione et al. *fAux: Testing Individual Fairness via Gradient Alignment.*

$$\textbf{Gradient Alignment}: grad(x) = |\Delta_{M(x)}(\Delta^{\top}_{M_{aux}(x)}\Delta_{M_{aux}(x)})^{-1}\Delta^{\top}_{M_{aux}(x)}|_{\infty}$$

Figure 3.1 Displays a toy example that illustrates gradient alignment. At points 1, 3, and 4 the gradients are disjoint or small, showing high local independence and, consequently, local fairness. Conversely, the gradients of the two models with regard to point 2 are parallel with similar magnitude, showing a low level of local individual fairness.

The gradient alignment value is used to identify examples that display the greatest individual fairness violations. This set of points that are treated unfairly can be used to detect the sources for this unfairness by viewing what the target and auxiliary models base their predictions on over this subset.

3.3.2 Identifying Proxy Features

Proxy features are detected over the subset of the data with the highest individual fairness violations, as detected by gradient alignment. The proxy identification method can then highlight the set of features that are conveying the most information about the protected feature. I have developed three methods for ranking features for removal, each with different prioritization for removal according to the natural relationship between fairness and performance.

Fairness / Performance Tradeoff

Features often contain information about both the primary attribute and the protected attribute. As a consequence, when we identify a proxy feature and remove it from the prediction process, we remove useful information about the primary task and decrease our predictive performance. When we identify proxy features, it is not quite as simple as identifying features that are exclusively informative about the protected attribute, we must also consider that we are removing information about the primary attribute and harming our predictive ability.

The first two methods are based on feature importance estimations, which we base on Shapley value estimations. Classic Shapley regression values are intended for linear models, where the values represent feature importance. Values are calculated by retraining models on every subset of features $S \subseteq F$ and valuing each feature based on the prediction values on models with that feature and without. This method requires significant retraining but also requires at least $2^{|F|}$ separate models to cover all combinations of included features, making it intractable for even moderate feature sizes. Methods to approximate the Shapley values by iterating only over local feature regions, approximating importance using samples from the training dataset, and other approaches have been proposed to reduce computational effort. The chosen platform for this work, DeepSHAP [27], is an efficient Shapley value estimation algorithm for DNNs. DeepSHAP uses linear composition rules and backpropagation to calculate a compositional approximation of feature importance values. DeepSHAP uses a set of background samples for reference, and these background samples are randomly sampled from the high alignment samples.

To this end, the following methods prioritize removal based on different criteria, giving the model owner control over prioritizing model performance or fairness.

Ratio: $ratio(feat) = \frac{importance(feat, M_{aux})}{importance(feat, M)}$

For each feature $feat$, we calculate the importance to the auxiliary model M_{aux} as well as the importance to the target model M and then calculate the ratio $\frac{importance(feat, M_{aux})}{importance(feat, M)}$. This selection method prioritizes features that are most important for the auxiliary task, while attempting to minimize the importance for the primary task. In this manner it attempts to prioritize feature removal on the ones that can both increase fairness as well as maintain a high model performance.

Product: $prod(feat) = importance(feat, M_{aux}) \cdot importance(feat, M)$

The product method uses the importance to both the target model and the auxiliary model to rank features based on overall importance. The features that are ranked the highest are the ones that convey the most information about the auxiliary task while also being very important to the primary task. This method greatly improves fairness, but can come with significant performance degradation.

Mutual Information

If a selection method grounded in information theory is desired, we can rank features by the mutual information between a feature and the protected feature $M(feat, c) = H(feat) - H(feat|c)$, where H is a measure of entropy. This method provides an estimation of the most likely proxy features based on information theory, under the assumption that the feature that carries the greatest amount of mutual information with the protected feature is a likely proxy feature. However, this method does not measure the importance of any feature to the target model, and could identify a proxy feature that is not used in any predictive capacity.

Feature Ranking

Table 3.1 shows the results of each of these ranking methods on a model trained on the Adult dataset, where the primary attribute is *income* and the protected attribute is *gender*. In each instance the ranking method has identified the *relationship* feature as a proxy feature. The relationship feature contains gendered categories such as *husband* and *wife*, and thus is heavily correlated with the protected feature. We can also see the difference in priority between the ratio and product selection methods, where the product selection places a significant emphasis on *occupation* because it also contains gendered information, whereas the ratio selection prioritizes it lower because it also contains a large amount of information about the primary attribute *income*.

Fair Models

Once we have identified the proxy features and ranked them in order of performance and fairness requirements, we can perform our model repair to mitigate the impact of these features. We can evaluate model fairness through the gradient alignment metric, and we

Rank	**Ratio**	**Product**	**Mutual Info**
1	relationship	relationship	relationship
2	race	occupation	race
3	workclass	race	country
4	occupation	capital.gain	occupation
5	hours.per.week	education	hours.per.week
6	marital.status	hours.per.week	workclass
7	education	age	age
8	country	workclass	marital.status
9	capital.loss	marital.status	education
10	capital.gain	capital.loss	capital.gain
11	age	country	capital.loss

Table 3.1: Feature rankings for each method on the Adult dataset

can posit that when we compare two models, the model with a lower gradient alignment over a set of samples is the fairer model over that set of samples. We can thus evaluate our feature selection methods on the dual objectives of predictive performance and gradient alignment.

To evaluate these methods, we train a *fair model* M_{fair} on data without the identified proxy features. Because the fair model has never learned from these features, it cannot leverage the information from them, and should have both a reduced model performance and reduced gradient alignment compared to the original model M.

Figure 3.2 shows a comparison of the gradient alignment of M and M_{fair}, both with regard to the auxiliary model M_{aux}. The fair model, which has had two features identified as proxy features and removed, has a mean closer to 0 and lower standard deviation than the target model. This signifies that the fair model displays a significant improvement in individual fairness compared to the target model. However, this fairness improvement does come with a small loss in accuracy.

3.3.3 Model Repair

Our goals for model repair are twofold. First, we want any repair method to result in similar performance to the fair model that's never seen the proxy features, both in terms of predictive accuracy and fairness. Second, we want our repair method to require less resources than simply training a fair model.

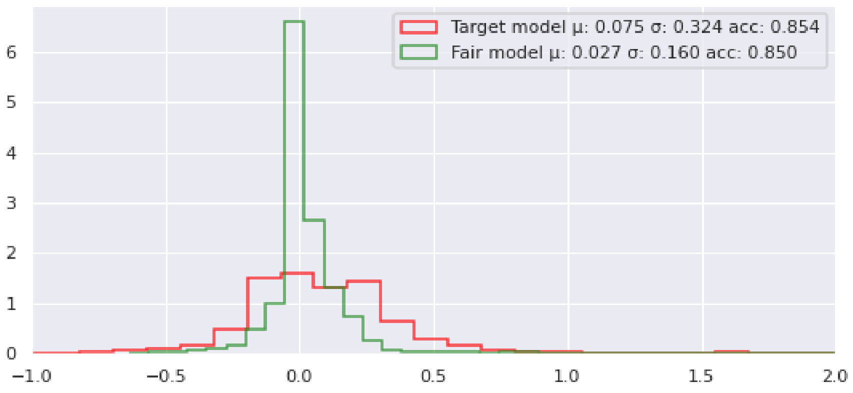


Figure 3.2: Histogram displaying the gradient alignment of target model and fair model

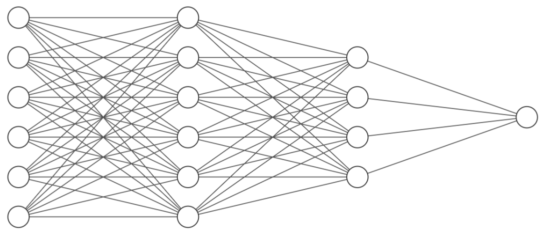
We can leverage the prior learning from the model to achieve these goals. We do so by removing the input neurons corresponding to features we wish to remove from the network entirely, along with all corresponding weights. This alone is sufficient to make the model behave fairly, but biases and other weight parameters have learned in a manner that assumes those features are present, so some fine tuning is necessary to restore model performance.

Figure 3.3 shows a simplified example of feature removal. The nodes corresponding to the protected feature are identified and those nodes along with connected weights are removed from the model, leaving a repaired model without those weights. The subsequent fine tuning results in other weight and bias values that reflect the reduced information the model must now make predictions from.

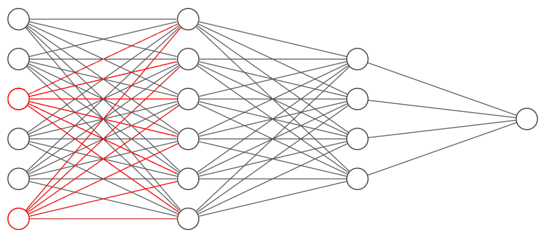
This model repair leverages concepts from transfer learning [31] to efficiently utilize prior learning in order to avoid a full retraining cycle. In practice, this results in a repair method that requires little computational time and few learning cycles to product a similar model to one that has been fully trained without the proxy features.

3.4 Empirical Evaluation

I performed an evaluation of the proxy feature identification methods for model fairness and of and the repair method. The model fairness and repair results are compared against a baseline of randomly selecting features for removal, and we compare all results on the



(a) Example DNN



(b) Proxy feature input nodes and weights highlighted

Figure 3.3: An example of the basic DNN repair method. Proxy feature input nodes and connected weights have been identified and marked with red and will be removed.

dual goals of model performance and individual fairness. In each instance the results show improvement over the baseline, and the repair method effectively produces a similar model to the trained-from-scratch model, but requires significantly less resources.

Experimental Environment: All algorithms are implented in Python 3.7 and use the PyTorch deep learning library [32]. Data processing is done with the pandas data analysis library, and feature importance calculations are done using the SHAP DeepExplainer and the Scikit-Learn Mutual Information libraries. All experiments were ran on Google Colab notebooks.

Dataset: We carried out an evaluation using the UCI Machine Learning Repository's Adult dataset [8]. This dataset consists of 32,561 records with 15 features including the target feature (income $\geq \$50,000$) and sex. The "marital.status" feature was condensed into a binary feature by combining 'Never-married', 'Divorced', 'Separated', and 'Widowed' into a "single" category and combining 'Married-civ-spouse', 'Married-spouse-absent', 'Married-AF-spouse' into a "married" category. The "fnlwgt" and duplicate numerical "education" feature "education.num" were dropped. Remaining features were either normalized around mean 0 and standard deviation 1 (for numerical features) or encoded into one-hot binary values (for categorical features). Table 3.2 summarizes this processing.

The result is a dataset with 32,561 rows and 100 columns representing 11 input features, one target feature, and one protected feature. The data was split into stratified training and test sets, and those were used to produce training and test datasets for both income prediction and gender prediction.

Neural Network: The DNN used for training is an MLP with 2 hidden layers with width 256 and 64, ReLU activation functions, and a softmax output layer. Dropout regularization was used during training, and the MLP was optimized for negative log-likelihood using the

Feature	**Type**	**Representation**
age	Numerical	Normalized float
capital.gain	Numerical	Normalized float
capital.loss	Numerical	Normalized float
hours.per.week	Numerical	Normalized float
finalwgt	Numerical	N/A
education.num	Categorical	N/A
education	Categorical	One-hot binary values
workclass	Categorical	One-hot binary values
occupation	Categorical	One-hot binary values
relationship	Categorical	One-hot binary values
race	Categorical	One-hot binary values
native.country	Categorical	One-hot binary values
Marital Status	Categorical	Binary value
income $\geq \$50,000$	Categorical	Binary value (target)
sex	Categorical	Binary value (protected)

Table 3.2: Features for Adult dataset

Adam optimizer. Identical architecture is used for the auxiliary model.

Results: All experiments were replicated 10 times and the presented results are either a randomly sampled result or the mean values over the 10 runs. The target, auxiliary, and fair models were each trained for 20 epochs. All results were generated over the test datasets.

3.4.1 Proxy Feature Identification

The proxy identification methods are compared against a a baseline of randomly sampling a feature to remove. The results are presented in figure 3.4. While all 3 methods outperform the baseline, the ratio method improves fairness while still maintaining model efficacy, compared to the product method that greatly improves fairness at the cost of reduced model performance. This underscores the responsibility model owners have to choose the ideal method for their needs.

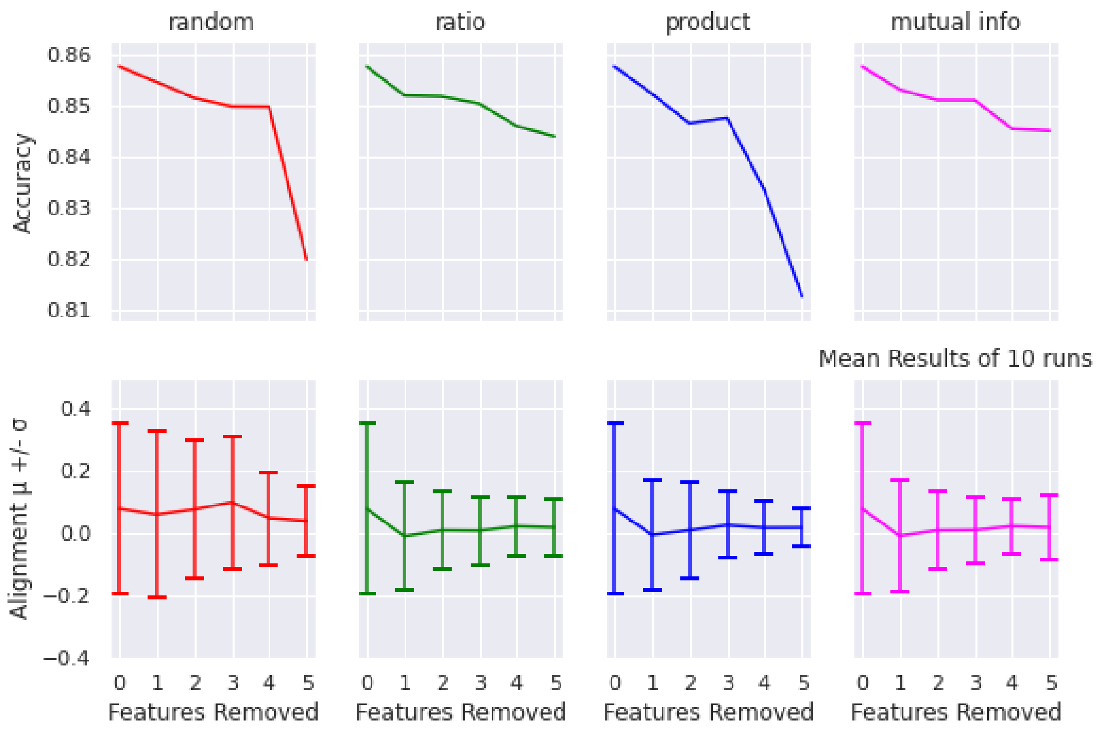


Figure 3.4: Results of proxy feature selection methods. The top row shows the model accuracy as features are removed, while the bottom row shows the mean and standard deviation of the gradient alignment values over the same removed features.

3.4.2 Model Repair

To evaluate model repair, a fair model was trained for 20 epochs without the removed features, and the original model was repaired and fine tuned for 2 epochs. Figure 3.5 displays histograms showing the gradient alignment values over the test dataset for each method. In each, the results for the repaired model and the fair model are very similar both in mean, standard deviation, histogram shape, and model accuracy. Figure 3.6 shows the same histogram results for removal of 3 features, and in each method we see the repair method is as effective as training a fair model from scratch, yet requires only 1/10 the training iterations.

3.4.3 Discussion

This evaluation shows significant improvement in individual fairness with regard to a protected feature. However, while this method is both effective and efficient, it is unclear if it is optimal.

It's possible that some method exists that can remove the effect of proxy features without removing the features themselves, due to the fact that proxy features convey meaningful information about both the target feature and the protected feature. Some more sophisticated method may be able to extract these, separating the feature into target information and protected information. To this date, I am unaware of any method that claims to do so, and without access to the ground measurements that compose x from latent representations z and z_c it is impossible to invert the feature composition function $x = f_g(z, z_c)$. Without this decomposition, however, this method significantly improves fairness through feature removal.

Additionally, it's possible that the repair method could be improved to not require fine-tuning. We are not aware of any methods that can repair a model in such a direction without a fine-tuning step to restore performance, but it is possible that some exists and future research could focus on determining a mathematical or algorithmic method for fine-tuning free repair.

The results as presented, although open to improvement, show that individual fairness violations can be diagnosed and repaired efficiently.

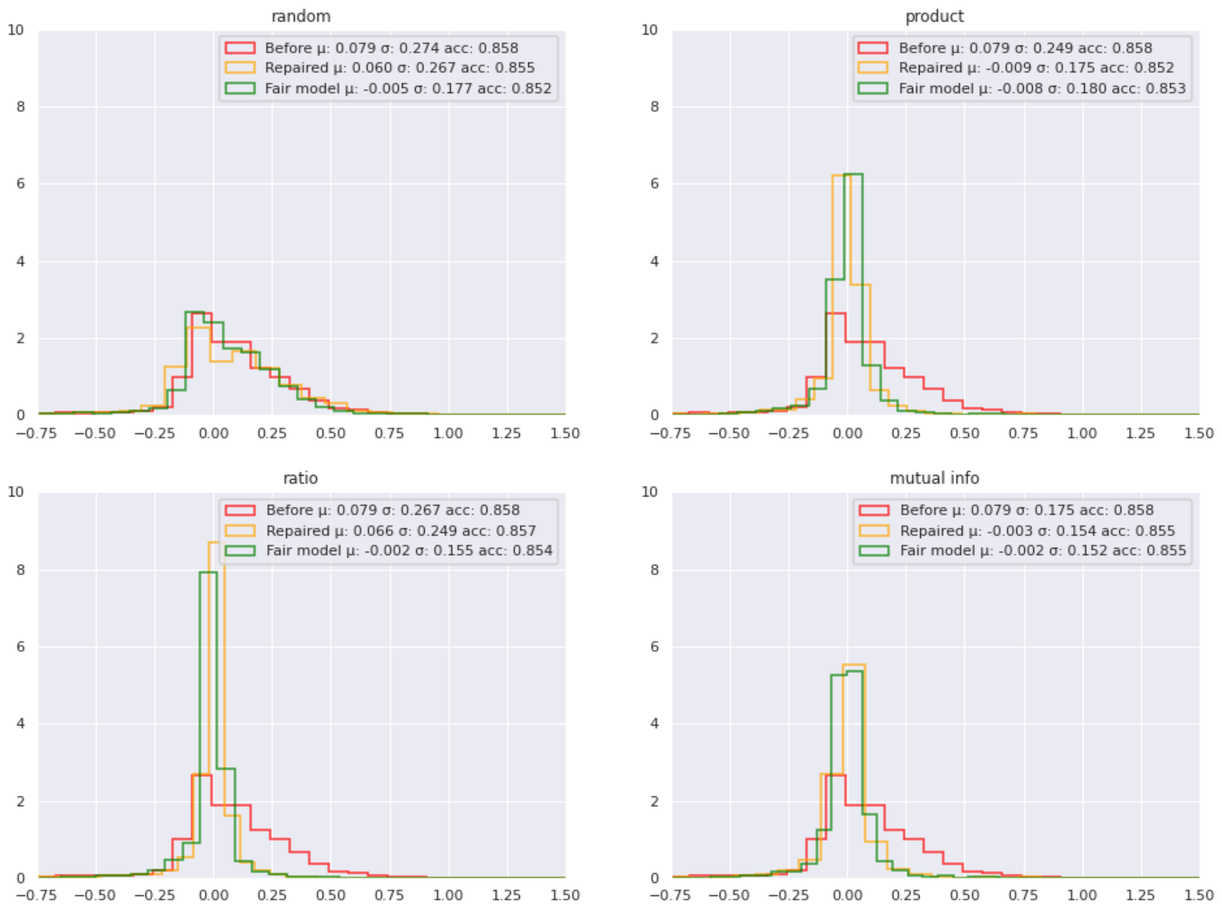


Figure 3.5: Histograms showing the gradient alignment values for the original model, repaired model, and fair model with 1 feature removed.

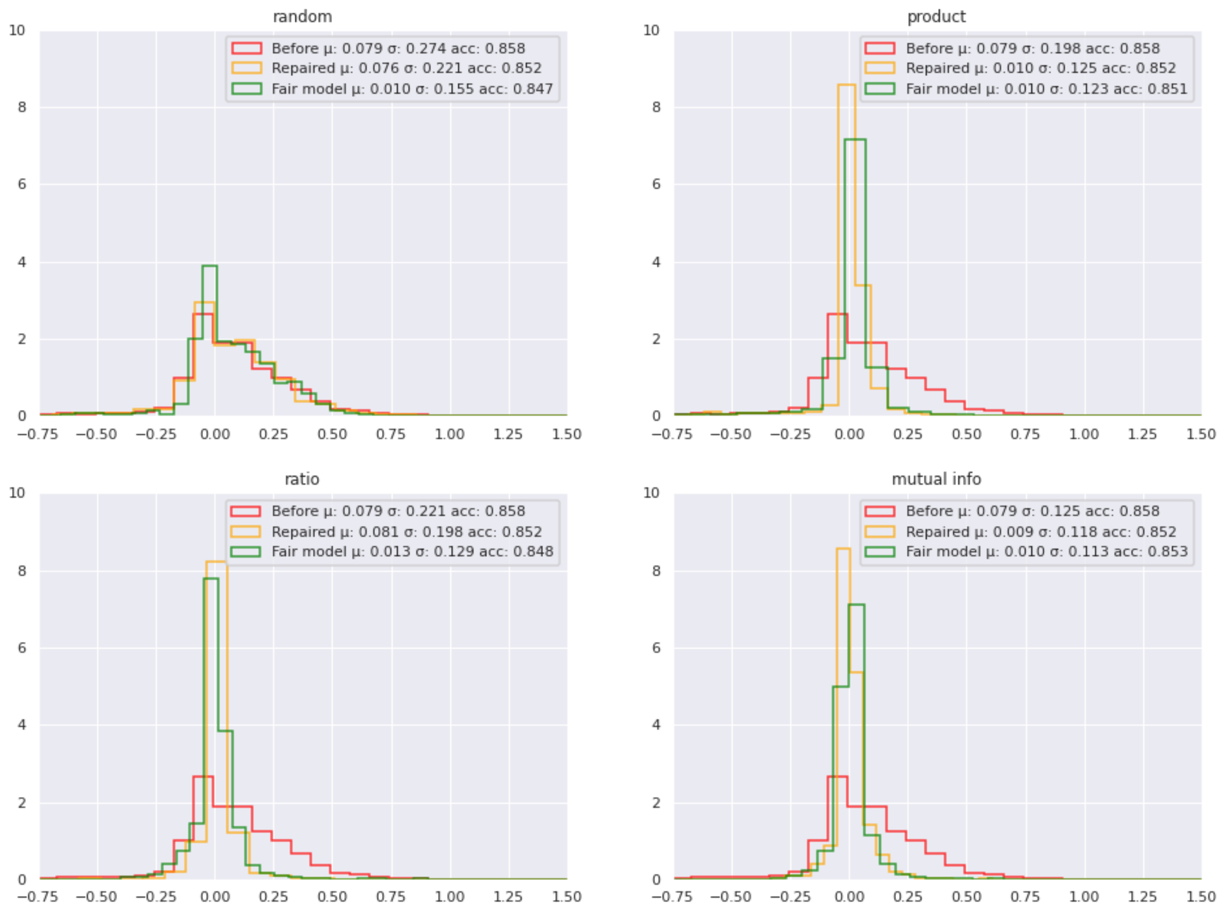


Figure 3.6: Histograms showing the gradient alignment values for the original model, repaired model, and fair model with 1 feature removed.

3.5 Conclusion

Algorithmic bias and fairness is a topical and extremely important issue for any entity using DNNs for decision making. In this chapter I presented an end-to-end repair method that detects individual fairness violations, diagnoses the features most responsible for these violations, and repairs the model to remove the influence of these features. This method gives the model holder flexibility for selecting features to remove based on desired fairness and performance priorities and runs in a fraction of the runtime compared to training a new model from scratch.

Despite these advances, this is a relatively new and constantly expanding field. These results, more than anything, highlight the need for further inquiry into fairness metrics and ways to evaluate and diagnose unfair predictions. Unfair behavior can be subtle and insidious, and the near ubiquity of machine learning decision making tools underscores the importance of making sure those predictions help repair demographic biases and disparate treatment rather than deepen those issues.

Chapter 4

Conclusion

State-of-the-art machine learning models increasingly require huge amounts of data and training time - the popular language model GPT-3 took over 3000 petaflop/s days to train [3], an increase of over 600,000 times the computation required for the original AlexNet. These increases come with the need for efficient ways of modifying these networks after training is completed. This can take the form of transfer learning, robustness training, or any other form of manipulation. Regardless of the goal, the costly training time shows that retraining from scratch is insufficient. We need something better, and this thesis presents a small step forward in the subfield of neural network repair.

In this thesis I have presented two algorithmic approaches to neural network repair to solve two immediate issues. In the first, I address the problem of *unlearning*, that is, given a trained model, removing specific learned data from that model. The unlearning algorithms are then used to remove privacy-sensitive data from a trained model to protect user privacy against state-of-the-art data leaking attacks, in a manner that is significantly more efficient than simply training a new model. In the second, I use a novel individual fairness repair method to detect and remove features most responsible for individual fairness violations (referred to as *proxy features*) without needing to train a new, fair model from scratch. These somewhat disparate problems of machine unlearning for privacy and model repair for individual fairness both share a common thread, that of situations where a trained model can be modified to meet some desired property. In each of these cases model repair is more efficient than training a brand new model. The experimental results are excellent, and both show a great increase in suitability toward the desired goal with significantly less resources than retraining from scratch.

Despite the significant success of these methods, they are applied to specific domains for specific tasks and are certainly not universal. In each case, a requirement was identified and a method was determined to be effective at modifying a model to meet that requirement. The efficacy of these methods show that there is a future in this domain, but also highlights the need for greater understanding. We need greater understanding about how models retain information, how they make predictions, how the learning process modifies these properties, and so on. The inscrutability of models shows us that until we can shine a light into the opaque parts of the learning process, we can only take these kinds of steps. It underscores the need for provable, foundational mathematical understandings of neural networks and reinforces that a solid understanding of neural networks is the only way forward toward provable, guaranteed repair methods that are abstract enough for any desired task.

4.1 Future Work

Looking into the future, I believe the field of model repair will become increasingly important and widespread. It combines aspects of logical reasoning that is useful for encoding and expressing requirements, theoretical understanding of deep learning, industry applicability, and addresses the massive training time and data requirements of state-of-the-art models. In this new and growing field, this work is placed as a novel work that addresses two different ways of solving the problem of model repair. My work adds to this field in a way that establishes that model repair can be done in an efficient and effective way even on models that are industry scale (as opposed to formal logic methods that often fail to scale to even moderately sized networks, much less networks such as ResNET). Additionally, this work shows that one does not need to be satisfied with paradigms that require properties to be established before starting training: in these cases, and imaginably many others, models can be modified to suit requirements even after fully training a high performing model.

This field of model repair is not only expansive, but can encompass many fields that have significant research done. The field of model compression aims to take a trained model, retain the useful learning for the task at hand, and compress that information into a smaller and faster architecture. This is simply model repair for the purpose of inference efficiency. Approaches such as distillation [33] aim to compress models specifically through leveraging the positive aspects of learning. Robustness methods often use some sort of post-training method to ensure robustness, and this neatly fits into this conception of model repair. While model security is often done during or before the training cycle, or through some sort of data manipulation in the case of differential privacy or label softening [10, 9, 4, 1], it also often focuses on taking trained models and modifying them to be less vulnerable to some attack model. Regularization, one of the most foundational aspects of training a model, can be done on trained models to avoid overfitting. The concept of model repair is flexible and allows us to think of these tasks in terms of abstractions, and these abstractions allow us to draw connections between previously disparate fields, unifying research direction and prompting new ways of thinking about these concepts. This new way of analyzing these problems can hopefully lead to significant advances and cross-domain collaboration.

While we draw a distinction between model repair and active learning scenarios, model repair can give us new ways to think about tasks that were previously done using primarily active learning. Model robustness is often done through adversarial training [40], where a model is trained in an un-robust manner, and then the dataset is augmented with adversarial examples crafted from attacking that or other models. This active learning scenario iteratively increases the model's robustness to adversarial attacks by strengthening the weak points in the model's decision boundary with regard to adversarial examples. Model repair can let us instead think about robustness in terms of modifying (or augmenting) the trained model to be more robust against adversarial perturbations. An example of this could be randomized smoothing, where a model is made robust via modifying the architecture to take an ensemble of predictions over a set of inputs with Gaussian noise perturbations added, effectively countering ℓ_p norm based adversarial attacks without the need to train a new model [5]. The lessons we learn from active learning about what a desirable model looks like can potentially be applied to make an effective model repair method that does not require this active learning.

I want to note that this concept of model repair is not only already in use in machine learning research other than deep learning, but is easily done in many of these due to our greater understanding of these. In the problem of removing a datum from the learning from a model, a subset of data can be removed from a k-means clustering or linear regression model with simple mathematical operations [17]. The relative complexity and opacity of deep learning makes this much more difficult, but future advances could potentially lead to certified removal methods that can mathematically calculate how gradient steps may have been different had a certain datum not been present, propagating this calculation through the learning process to calculate exactly what the final parameters would be had that datum not been present. Although this is currently prohibitively difficult to calculate, advances in our understanding of deep learning will hopefully make calculations like this not only feasible, but practical. This would have significant implications for protecting against not only privacy-violating attacks, but also protecting against data poisoning, mislabeled training instances, or any other instances where a subset of training data leads to undesirable behavior.

As we learn more about how neural networks learn and what information they retain and use to make predictions, methods such as these will be able to get more sophisticated and handle all sorts of different required tasks. In the meantime, algorithms such as those presented in this thesis give a great amount of control over opaque and difficult to control traits such as privacy or fairness. The immediate benefit of these methods gives credence to their utility, and while further research into the unknowns continues, these advances can be made in parallel.

References

[1] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 464–473. IEEE, 2014.

[2] Lucas Bourtoule, Varun Chandrasekaran, Christopher Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning, 2019.

[3] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[4] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(Mar):1069–1109, 2011.

[5] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, pages 1310–1320. PMLR, 2019.

[6] European Commission. 2018 reform of eu data protection rules, 2018.

[7] Rachel Cummings and Deven Desai. The role of differential privacy in gdpr compliance. In *FAT'18: Proceedings of the Conference on Fairness, Accountability, and Transparency*, 2018.

[8] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[9] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[10] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[11] Cynthia Dwork, Adam Smith, Thomas Steinke, Jonathan Ullman, and Salil Vadhan. Robust traceability from trace amounts. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 650–669. IEEE, 2015.

[12] Lixin Fan, Kam Woh Ng, and Chee Seng Chan. Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks. In *Advances in Neural Information Processing Systems*, pages 4716–4725, 2019.

[13] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *ACM Conference on Computer and Communications Security*, 2015.

[14] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 17–32, San Diego, CA, August 2014. USENIX Association.

[15] Karan Ganju, Qi Wang, Wei Yang, Carl A Gunter, and Nikita Borisov. Property inference attacks on fully connected neural networks using permutation invariant representations. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 619–633. ACM, 2018.

[16] Antonio Ginart, Melody Guan, Gregory Valiant, and James Y Zou. Making ai forget you: Data deletion in machine learning. In *Advances in Neural Information Processing Systems*, pages 3513–3526, 2019.

[17] Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens van der Maaten. Certified data removal from machine learning models. *arXiv preprint arXiv:1911.03030*, 2019.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[19] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep models under the gan: information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 603–618, 2017.

[20] Nils Homer, Szabolcs Szelinger, Margot Redman, David Duggan, Waibhav Tembe, Jill Muehling, John V Pearson, Dietrich A Stephan, Stanley F Nelson, and David W Craig. Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays. *PLoS genetics*, 4(8):e1000167, 2008.

[21] Matthew Humerick. Taking ai personally: How the eu must learn to balance the interests of personal data privacy & artificial intelligence. *Santa Clara High Tech. LJ*, 34:393, 2017.

[22] Right to erasure, 2018.

[23] Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. High-fidelity extraction of neural network models. *arXiv preprint arXiv:1909.01838*, 2019.

[24] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *University of Toronto*, 2009.

[25] Matt J Kusner, Joshua R Loftus, Chris Russell, and Ricardo Silva. Counterfactual fairness. *arXiv preprint arXiv:1703.06856*, 2017.

[26] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. *Online Database*, 2010.

[27] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4765–4774, 2017.

[28] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Stand-alone and federated learning under passive and active white-box inference attacks. *arXiv preprint arXiv:1812.00910*, 2018.

[29] Laurens Naudts. How machine learning generates unfair inequalities and how data protection instruments may help in mitigating them. *R. Leenes, R. van Brakel, S. Gutwirth & P. De Hert (Authors), Data Protection and Privacy: The Internet of Bodies (Computers, Privacy and Data Protection)*, 2019.

[30] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4954–4963, 2019.

[31] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[32] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[33] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*, 2018.

[34] Yannik Potdevin, Dirk Nowotka, and Vijay Ganesh. An empirical investigation of randomized defenses against adversarial attacks. *arXiv preprint arXiv:1909.05580*, 2019.

[35] Cynthia Rudin, Caroline Wang, and Beau Coker. The age of secrecy and unfairness in recidivism prediction. *arXiv preprint arXiv:1811.00731*, 2018.

[36] Noel E Sharkey and Amanda JC Sharkey. An analysis of catastrophic interference. *Connection Science*, 1995.

[37] James A Sherer. When is a chair not a chair? big data algorithms, disparate impact, and considerations of modular programming. *Computer and Internet Lawyer*, 34(8):6–10, 2017.

[38] Yi Shi, Yalin Sagduyu, and Alexander Grushin. How to steal a machine learning classifier with deep learning. In *2017 IEEE International Symposium on Technologies for Homeland Security (HST)*, pages 1–5. IEEE, 2017.

[39] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017.

[40] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

[41] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pages 601–618, 2016.

[42] Michael Veale, Reuben Binns, and Lilian Edwards. Algorithms that remember: model inversion attacks and data protection law. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 376(2133):20180083, 2018.

[43] Eduard Fosch Villaronga, Peter Kieseberg, and Tiffany Li. Humans forget, machines remember: Artificial intelligence and the right to be forgotten. *Computer Law & Security Review*, 34(2):304–313, 2018.

[44] David West. Neural network credit scoring models. *Computers & Operations Research*, 27(11-12):1131–1152, 2000.

[45] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pages 268–282. IEEE, 2018.

[46] James Zou and Londa Schiebinger. Ai can be sexist and racist—it's time to make it fair, 2018.