Deep Reinforcement Learning for Data-Agnostic Post-Training Debiasing of Black-Box Machine Learning Models

Thomas Václav Pinkava

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science

University of Washington

2024

Committee:
Afra Mashhadi
Michael Stiber
Erika Parsons

Program Authorized to Offer Degree:
Computer Science and Software Engineering

University of Washington

Abstract

Deep Reinforcement Learning for Data-Agnostic Post-Training Debiasing of Black-Box Machine Learning Models

Thomas Václav Pinkava

Chair of the Supervisory Committee:
Afra Mashhadi
Department of Computing and Software Systems

As reliance on Machine Learning systems in real-world decision-making processes grows, ensuring these systems are free of bias against sensitive demographic groups is of increasing importance. Existing techniques for automatically debiasing ML models generally require access to either the models' internal architectures, the models' training datasets, or both. In this paper we outline the reasons why such requirements are disadvantageous, and present an alternative novel debiasing system that is both data- and model-agnostic. We implement this system as a Reinforcement Learning Agent and employ it to debias four target ML model architectures over three datasets. Our results show performance comparable to data- and/or model-gnostic state-of-the-art debiasers.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

This work would not have been possible without the enthusiastic support of my advisor, Dr. Afra Mashhadi, and the members of her lab, off of whom many of the ideas discussed herein were bounced.

I would also like to thank my brother Edward for the generous loan of a second computer, without which the following experiment would probably still be running.

Chapter 1

INTRODUCTION

Machine-Learning (**ML**) systems are being increasingly used in real-world decision-making processes, such as employment offers, credit evaluations, or predictions of criminal recidivism. Researchers have often found that these decision-making algorithms exhibit bias towards sensitive demographic groups, and have raised arguments in favour of reducing this bias since Machine Learning was far less prevalent than it is today [8, 3, 6, 11, 38, 39].

How best to reduce the real-world undesirable effects of a biased Machine Learning model is an open philosophical question [18, 10, 39]. Our work is primarily concerned with developing a practical mechanism, so we shall merely skim the quagmire to obtain a definition for our problem.

Friedler et al. [15] define two 'worldviews' of machine-learning fairness, based on the correspondence between *observation space* (the data consumed by our ML model) and *construct space* (the 'reality' we wish to predict). The former ('What You See Is What You Get') assumes that the observation space accurately represents the construct space (the example they use is SAT scores being a measure of intelligence), and that fair decisions can be made directly from observations, often called *individual fairness*. The latter ('We're All Equal') assumes that all individuals are similar in the construct space, and that discrepancies between construct and observation arise from systemic bias (the example being that SAT scores are influenced by the SAT preparation one can obtain, rather than native intelligence). The existing techniques discussed in this thesis, including the benchmarks to which we compare our results and our own introduced technique, fall squarely into the latter paradigm, which is termed *group fairness*.

Debiasing ML models to obtain group fairness entails the adjustment of a model's predictions with respect to *sensitive attributes*; that is, features of demographic data representing certain class memberships that affect real-world observable disparities, such as race or sex. Such debiasing techniques often operate on tabular demographic data, as this data readily presents sensitive attributes over which fairness can be enforced. Our thesis remains within this domain; we discuss expanding to alternative data modalities in Section 7.3. The mechanisms for obtaining a measure of bias from the machine's predictions with respect to sensitive attributes are discussed in greater detail in Chapter 3.

Existing group-fairness-based debiasing techniques share certain common drawbacks. Some, like the *fairness-accuracy tradeoff* [40], are unavoidable consequences of debiasing. Others may be resolvable. In this Thesis we identify two practical properties found in existing debiasers (**model-** and **data-gnosticity**), make an argument for their obviation, and present a means by which this may be effected.

In Chapter 2, we summarize existing debiasing techniques, and identify the shortcomings that we wish to address, from which we distill a list of five requirements for a proposed novel debiasing technique (Section 2.3). In Chapter 4, we show a practical implementation of a technique that meets those requirements. In Chapter 5, we outline an experiment to demonstrate the efficacy of our new system; the results of that experiment are given in Chapter 6; we discuss the implications of our work and future directions for research in Chapter 7.

Chapter 2

BACKGROUND

2.1 Existing Debiasing Techniques

In this chapter, we outline existing debiasing techniques, describe their common shortcomings, and propose an alternative approach to debiasing that addresses these problems. Barocas et al. [4] divide extant automatic debiasing techniques into three classes, based on where they appear in the Machine Learning pipeline:

2.1.1 Pre-process techniques

Pre-process techniques are employed before the ML model is trained, and act by modifying the training data in some fashion. These modifications are usually effected by another model trained to perform them.

Such a model might learn a direct data-transformation, as in the work of Calmon et al. [9], or might detect and remove the data features most responsible for bias, as in the work of Li et al. [25], Lum and Johndrow [30], or Kamiran and Calders [21]. The latter also investigate weighting the features to mitigate bias, as do Mehrabi et al. [32], who use an attention-based preprocessor.

Another possibility is modifying the data representation. Locatello et al. [29] provide a summary of this process: the aim is to convert the data to a format that improves the fairness of models trained on it. Zemel et al. [47] present such a technique; while Madras et al. [31] offer an adversarial variant.

One might also add or remove whole data items to mitigate bias; Dixon et al. [12] offer a method involving adding synthetic data to the dataset, while Kamiran and Calders [21] also describe a resampling technique.

2.1.2 In-process techniques

Another avenue for reducing the bias of a model occurs during the model's training; such techniques are termed *in-process*.

Such techniques often consist of changes to the model's optimization system; Agarwal et al. [1] modify a model's cost-sensitivity to include bias, while the work of both Zhang et al.(2018) [49] and Yang et al. [45] add an adversarial module which attempts to reconstruct the sensitive attribute(s) from the predictions, and whose success is included in the model's loss function. We experimented with adapting such a system for our needs, and implemented Yang et al.'s algorithm; we use our implementation as a benchmark, which is discussed in Section 5.2.

Another approach is to adjust properties of the training process. F. Cruz et al. [13] present a fairness-aware hyperparameter optimizer, and Nguyen et al [34] add a fairness term to AutoML, an automated model development pipeline.

If the target model uses encodings or latent-space representations, one can also modify these to improve fairness. Zhang et al.(2023) [50] provide a debiaser using latent-space perturbation, while Zeng et al. [48] do the same with an adversarial system. Bolubaski et al. [8] do the same in language modelling, with a technique for debiasing word embeddings through geometric modifications.

2.1.3 Post-process techniques

Post-process techniques are applied after the model is trained, and are divided into two subtypes.

The former subtype manipulates the structure of the trained model; such techniques need only be applied once, and produce a model with reduced bias, as in the work of Woodworth et al. [43] and Fish et al. [14].

The latter subtype involves adjusting the output of the model at prediction time. These techniques generally take the form of another model attached to the target model output,

which has been trained to reduce bias by modifying predictions. Alabdulmoshin and Lucic [2], Kamiran et al. [22], and Wei et al. [42] all present variants thereof.

2.2 Drawbacks of Existing Techniques

The above three classes of debiasing techniques each entail certain drawbacks.

Pre-process approaches require forethought; as they must be applied before the model is trained, fairness must be a consideration from the beginning of the modeling process. As a corollary, if the model is later found to be biased, to recover fairness using pre-process techniques would require a complete retraining of the model. Additionally, the model cannot be so debiased without access to the training data, which may be restricted due to security, ownership, privacy, or resource-management concerns.

In-process techniques likewise require the consideration of the training party, and the modifications to the training process, unless well-defined, well-publicized, and generalizable, are likely to be practically inapplicable, unfamiliar to the trainer, and/or expensive. Debiasing a trained model with in-process techniques also entails retraining.

Post-process techniques place the least burden on the training party, and are therefore suitable for post-factum application to models found to exhibit bias. However, they too have drawbacks: techniques that modify the model's architecture require a white-box target, which may not be possible if the model is proprietary, secret, or otherwise inaccessible. Post-processing the model's output incurs a complexity cost, as the post-processor is an additional step in the model's inference process. Such output transformations, being unaware of data and model, are also susceptible to poor performance (Woodworth et al. [43]).

Many of the works discussed above attempt to ameliorate the shortcomings of their class by providing generalizability guarantees. Calmon et al. [9] and Lum and Johndrow [30], whose techniques are pre-process, both claim to work well with any data modality. Agarwal et al. [1]'s in-process technique can debias arbitrary black-box target models. Kamiran et al. [22]'s post-processor is capable of both operating on a wide variety of data and a wide variety of models, and since, as they state, "retrained classifiers can be made discrimination-

aware at prediction time" the requirement of a cooperative training party is relaxed. Their system, however, does take the form of an extra component that must process every prediction, and thus increases overall model complexity and execution time.

Table 2.1: Summary of Discussed Extant Techniques, grouped by debiasing phase

Work	Year	Type	Stage	Model-agnostic	Data-agnostic
Calmon et al. [9]	2017	Data transformation	Pre-	Yes	No
Dixon et al. [12]	2018	Data-aware synthesis	Pre-	Yes	No
Kamiran and Calders [21]	2012	Feature weighting	Pre-	Yes	No
Li et al. [25]	2022	Feature culling	Pre-	Yes	No
Locatello et al. [29]	2019	Representation	Pre-	Yes	No
Lum and Johndrow [30]	2016	Feature culling	Pre-	Yes	No
Madras et al. [31]	2018	Representation	Pre-	Yes	No
Mehrabi et al. [32]	2021	Attention preproc	Pre-	Yes	No
Zemel et al. [47]	2013	Representation	Pre-	Yes	No
Agarwal et al. [1]	2018	Cost adjustment	In-	No	No
Biswas and Rajan [7]	2020	Adversarial	In-	No	Yes
Bolukbasi et al. [8]	2016	Latent adjustment	In-	No	No
F. Cruz et al. [13]	2021	Hyperparameter optimization	In-	No	Yes
Liao and Naghizadeh [26]	2023	Cost adjustment	In-	No	No
Nguyen et al [34]	2023	Fair-aware AutoML	In-	No	Yes
Su et al. [36]	2023	Bias term in loss	In-	No	Yes
Wadsworth et al. [41]	2018	Adversarial	In-	No	Yes
Yang et al. [45]	2023	Adversarial	In-	No	Yes
Zeng et al. [48]	2022	Latent adjustment	In-	No	No
Zhang et al. [49]	2018	Adversarial	In-	No	Yes
Zhang et al. [50]	2023	Latent adjustment	In-	No	No
Alabdulmoshin and Lucic [2]	2022	Prediction adjuster	Post-	Yes	No
Fish et al. [14]	2016	Structure adjuster	Post-	No	Yes
Kamiran et al. [22]	2012	Prediction adjuster	Post-	No	Yes
Wei et al. [42]	2021	Prediction adjuster	Post-	No	No
Woodworth et al. [43]	2017	Structure adjuster	Post-	No	Yes
Zafar et al. [46]	2019	Decision-boundary modification	Post-	No	Yes
This Thesis	**2024**	**RL Data synthesis**	**Post-**	**Yes**	**Yes**

2.3 Proposed New Method

2.3.1 Desiderata

The shortcomings discussed above reveal a hitherto-unfilled niche in machine debiasing, which we can circumscribe with the following desiderata:

1. *Data-Agnosticism*: To avoid the need to access data that may be private, secret, proprietary, etc., the debiasing system should not require access to the target model's training data. Ideally, no provision should need to be made for what the target's featurespace actually corresponds to.

2. *Model-Agnosticism*: To avoid the need to perform expensive, complex, or impossible modifications to the target model, the system should not be able to read from or write to the target model's structure, weights, architecture, etc. By extension this implies that the system need not be concerned with said structure, weights, architecture, etc.

3. *Applicable After Training*: To avoid the need for a fairness-aware training party, the system should be usable on targets that have already been trained.

The work of Kamiran et al. [22] meets the above, but, since it requires appending a component to the model's output, suggests a further two desiderata:

4. *No complexity increase*: We do not wish to increase model size or execution time, or to add unwieldy components; we may also wish to debias models that are lightweight. We therefore want a system that adds no components to the target model.

5. *Single-pass post-factum training*: We do not wish to burden the end user of the model with the responsibility of keeping it unbiased. The debiasing system must therefore be a task that, once accomplished, produces a debiased model, and not an ongoing process.

2.3.2 Process Overview

The system desiderata give us some notion of what manner of thing we endeavour to create. The requirements that it be data- and model-agnostic suggest that the system be itself a machine learning model, whose training takes place 'on-line', that is, only once it is presented with a target model to debias.

The requirements also show how the target model must be debiased: since we cannot perform alterations to a known data stream, either entering or leaving the target, or modify the weights or architecture of the target model itself, and we still wish to debias by modifying the target, the only avenue left is to hijack the target's training procedure. Our debiasing model can feed the target model synthetic training data generated specifically for the purpose of reducing the target's bias, data whose format is derived solely from the input shape of the target and whose content is unconcerned with the real-world meaning of any of its features.

This interaction mode has already been explored to some degree. Xiao et al. [44] measure the fairness of a target model by presenting it with adversarially generated examples designed to maximize the exposure of the target's potential biases. Zeng et al. [48] go further towards our goal, with a system that adversarially generates synthetic training data; however, their system is limited to the domain of face attribute classification. We aim to go further, to produce the data required to debias any target on-the-fly.

Chapter 3
MEASURING BIAS

As discussed in Chapter 1, group fairness in ML is defined by the relationship between predictions and sensitive demographic attributes. Debiasing a model automatically, therefore, entails minimizing some metric derived from the relationship between the model's input, the sensitive attributes therein, and the model's output. Fairness research employs a great variety of these [39]. As our work is primarily concerned with prototyping a novel system we will not be delving into the active debate surrounding the relative merits of the panoply of available metrics. We will also not be considering multiple fairness metrics simultaneously; Teodorescu and Yao [40] show that said problem is computationally hard. We present below two possible metrics for our system to use.

3.1 Demographic Parity

Demographic Parity (called *Independence* by Barocas et al. [4]) is a measure of a classifier output's statistical independence from the sensitive attribute(s). Demographic Parity is satisfied if the following condition holds for sensitive attribute(s) A and predictions $\hat{Y}$:

$$\forall a, b \in A, \;\; \hat{y} \in \hat{Y} : \quad P(\hat{Y} = \hat{y} | A = a) = P(\hat{Y} = \hat{y} | A = b) \tag{3.1}$$

Our initial experiments used Demographic Parity as their bias criterion. It has certain shortcomings as a bias metric, however. Requiring statistical independence of a predictor's output from the sensitive attribute(s) is tantamount to enforcing outcome equity, which does not consider whether the individuals classified actually possess the quantity being predicted (note that Eqn. 3.1 does not contain the ground-truth label(s) Y). We therefore opted instead to use Equalized Odds, defined below.

3.2 Equalized Odds

Equalized Odds is a metric proposed by Hardt et al. [17], and later generalized to multiclass problems by Woodworth et al. [43]. Equalized Odds improves on Demographic Parity by requiring that the predictor's output be independent of the sensitive attribute(s), but still conditional on the ground truth.

Consider, for instance, an example case of a machine-learning model designed to provide predictions in a job-offer process: we have a candidate pool with some sensitive demographic division, say, sex, and we wish to estimate each candidate's ability to do the job in question. If our machine-learning model respects Demographic Parity, its predicted qualified candidate set will contain an equal number of women and men; however, those candidates may be of wildly varying ability. Equalized Odds ameliorates this situation: the model is now required to accept qualified women at the same rate as qualified men, and reject unqualified women at the same rate as unqualified men. This may result in a different absolute number of women and men, but the model will not overestimate or underestimate the ability of one sensitive group relative to another.

Formally, Equalized Odds is satisfied if:

$$\forall a \in A,\ \ y, \hat{y} \in Y: \quad P_{\hat{Y}|AY}(\hat{y}|a, y) = P_{\hat{Y}|Y}(\hat{y}|y) \tag{3.2}$$

Our work considers the case in which both the sensitive attribute and the labels are binary, so the metric becomes:

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

Since we are training a machine learning model to optimize for bias reduction, it is necessary to convert the Equalized Odds metric into a scalar loss, which we (after Su et al. [36]) term the *Equalized Odds Violation* (**EOV**). The former of the above two equalities

represents parity between the True Positive Rate (**TPR**) and False Positive Rate (**FPR**)[1] between the sensitive and nonsensitive groups; to compute the EOV, we take the difference between those rates and select the maximum (i.e. worst) violation:

$$\begin{aligned} &\max_{y} \left| P(\hat{Y} = 1 | A = a, Y = y) - P(\hat{Y} = 1 | A = b, Y = y) \right| \quad y \in \{0, 1\} \\ &= \max(|\mathbf{TPR}_a - \mathbf{TPR}_b|, \quad |\mathbf{FPR}_a - \mathbf{FPR}_b|) \end{aligned} \tag{3.3}$$

The Equalized Odds Violation will be in the range $[0.0, 1.0]$, where the closer a model scores to zero, the less biased we consider it to be.

[1]It is sufficient to compare only TPR and FPR, as the other two cases, False Negative Rate (FNR) and True Negative Rate (TNR), are simply one minus TPR and FPR respectively.

Chapter 4
SYSTEM ARCHITECTURE

4.1 Task Overview

Section 2.3.2 outlines the device we wish to create: an ML model that reduces the bias of a target model by feeding it synthetically generated training data. Said model is not pretrained, as its task is specific to a given target model. We must now decide exactly how such a model is to be structured.

4.2 A Choice of Paradigm

4.2.1 Problems with the direct approach

Our system is, essentially, a machine learning model whose training process contains the training process of another machine learning model. For this reason a direct approach (generate synthetic data using the model, train the target on it, then measure the target's bias and backpropagate it through the model as loss) proves practically infeasible. We attempted to implement this direct strategy using the TensorFlow and PyTorch machine-learning libraries; both libraries presented different obstacles. A non-trivial training process in TensorFlow requires the use of a gradient-watching context ("GradientTape"). Running the target's backpropagation step means invalidating the GradientTape, meaning the gradients are no longer accessible at the overall model's backpropagation step[1]. In PyTorch, gradients are stored within the data tensors, and form linked chains in the forward pass, which backpropagation re-traverses; like TensorFlow's GradientTape, said backpropagation is destructive.

[1]In previous, unrelated work, we had circumvented this problem by exploiting the model's determinism: we generated the synthetic data *twice*, then fed one copy through the target, and backpropagated using the other. Obviously doubling the system's workload is undesirable, and there may have been assumptions about datapath integrity that the workaround violated; the model executed, but did not work.

Since we wished to use off-the-shelf machine-learning libraries, we therefore require a different paradigm, one that would allow us to isolate the target from the model and create an air-gap between the target's optimization step and the model's. One such suitable paradigm is *Reinforcement Learning*.

4.2.2 Reinforcement Learning

Reinforcement Learning (**RL**) is a Machine Learning paradigm in which an *agent* (the model) makes *observations* about an *environment*, then generates *actions*, which are given to the environment, producing *rewards*. The agent attempts to maximize these rewards by adjusting its actions, and thus learns through a trial-and-error process [20].

RL is useful to our work for a simple reason: it does not concern itself with the nature of the environment, only with its input and output; we can therefore 'hide' our target model's training process inside the environment. Our debiaser model therefore becomes an Agent, the synthetic training data it produces become analogous to actions, and the measurement of the target's bias after training on that data is the reward.

The remaining component of RL to incorporate are the observations. Initially, when we had realized that we would need the target's training data to measure bias, we relaxed the data-agnosticity of the model and devised an Agent that worked by data perturbation. The observations were data items from the target's initial dataset, and the Agent added perturbations to the data's features to form actions. This initial design, apart from undesirably cementing the system as reliant on data, failed to allow a useful element of RL Agents: the ability to predict future state. Many RL algorithms consider not only the reward for taking a given action having made a given observation, but also a discounted future reward for possible actions reachable from the resulting state. In our initial system, 'future state' was nonsensical: the Agent would be predicting the next data item it would see from the training set, which would be difficult and pointless.

We therefore rebuilt the Agent to generate its actions completely from scratch, and as observations we provided it with the target's current bias and the difference in bias from its

previous state incurred by the agent's previous action. This not only freed our Agent from requiring the target's dataset – thus reaffirming our model as data-agnostic – but made the Agent able to predict future state by estimating the bias of the target as the result of the action it took. This change was also one of the architectural changes we made that made the Agent efficacious.

4.2.3 Actor-Critic RL

Reinforcement Learning is one of the oldest subfields of ML, and it encompasses a wide array of techniques, so we have to choose a specific flavour best suited to our particular application. We chose to implement a variant of the Actor-Critic architecture, introduced by Sutton et al. [37] in 1999. In this setup, shown in Figure 4.1, the Agent is divided into two ML models.

The first, the *Actor*, models an *action policy*: a mapping from observations to actions, which provides the mechanism for the Agent's decision-making. The second, the *Critic*, models what is called the *Q-function*: a mapping from observation/action pairs to the expected reward for taking said action given said observation.

The key advantage of an Actor-Critic system is that the Actor is optimized to maximize the *expected* reward of its actions – which is to say it is trained by the Critic, and itself takes no information from the environment. This gap in the datapath is, as discussed above, necessary for our task.

4.2.4 Deep Deterministic Policy Gradient

We have another reason for choosing an Actor-Critic Agent, which stems from another requirement of our system. Many RL Agents are designed for discrete state and action spaces: the Agent's actions are chosen from a set of possibilities, and its observations are drawn from a number of possible states. A traditional Q-learner, for instance, models the Q-function using a table with a fixed number of rows (states) and columns (actions). Since our observation (the target's bias) is a continuous real number, and since our actions are

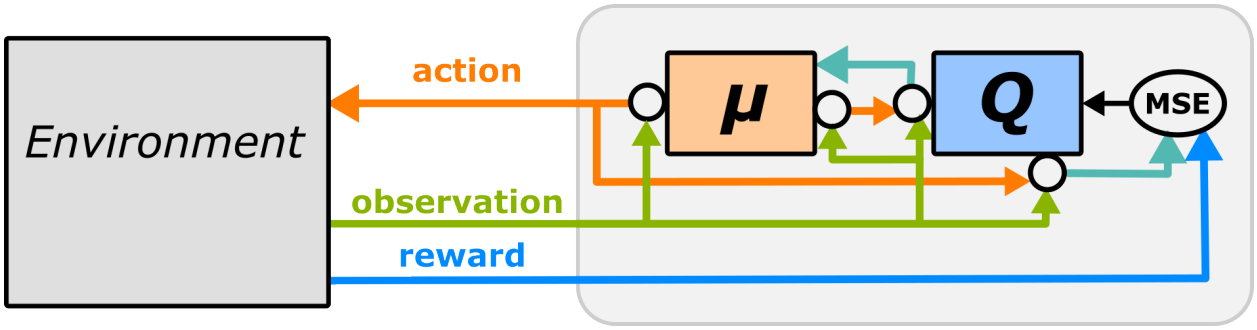

Figure 4.1: Actor-Critic RL Agent. Q denotes the Critic network, and μ the Actor network. The bubbles attached to each network denote a forward pass. The 'MSE' at left is the mean squared error between the Critic's prediction of the reward and the true reward, which forms the Critic's loss. The Actor's loss is derived directly from the Critic's predicted reward.

synthetic data items, that is, tensors of real numbers, we need an Agent capable of continuous action, such that it can generate synthetic data at arbitrary granularity.

We therefore draw our agent's theoretical groundwork and practical architecture from the work of Lillicrap et al. [27], who extend the Deterministic Policy Gradient learner of Silver et al. [35] into the continuous action space. G. Liu's PyTorch port [28] was used as a guide.

Lillicrap et al.'s implementation contains two refinements sometimes found in RL Agents, which we added to our own code, as both stabilize the Agent's training process:

1. The *Replay Buffer*: The Actor and Critic are trained on observation/action/reward/next observation tuples, so instead of generating these individually, passing them through the models individually, and running the model's optimizers, the tuples are placed in a FIFO queue of fixed size. At each training step, a sample of several tuples is drawn from the queue at random, and the Actor and Critic are trained thereupon. The buffer also provides a larger air-gap between the Target and Agent training processes.

2. *Secondary Actor and Critic*: To mitigate the volatility of the estimated future state,

the future-reward prediction is performed by a second Actor and Critic, who are not trained, but whose weights are soft-copied (their weights and the weights of the main Actor and Critic are mixed in some hyperparametric proportion) at each training step. Lillicrap et al. call these "target models," but this nomenclature of course clashes with our system's target model, so we term them "secondaries".

4.3 The Debiasing Process

Algorithm 1 shows the entire debiasing process, incorporating both the training of the Agent and the simultaneous debiasing of the target. Line numbers are referenced in the long-form description below. Figure 4.2 shows an overview of the datapath.

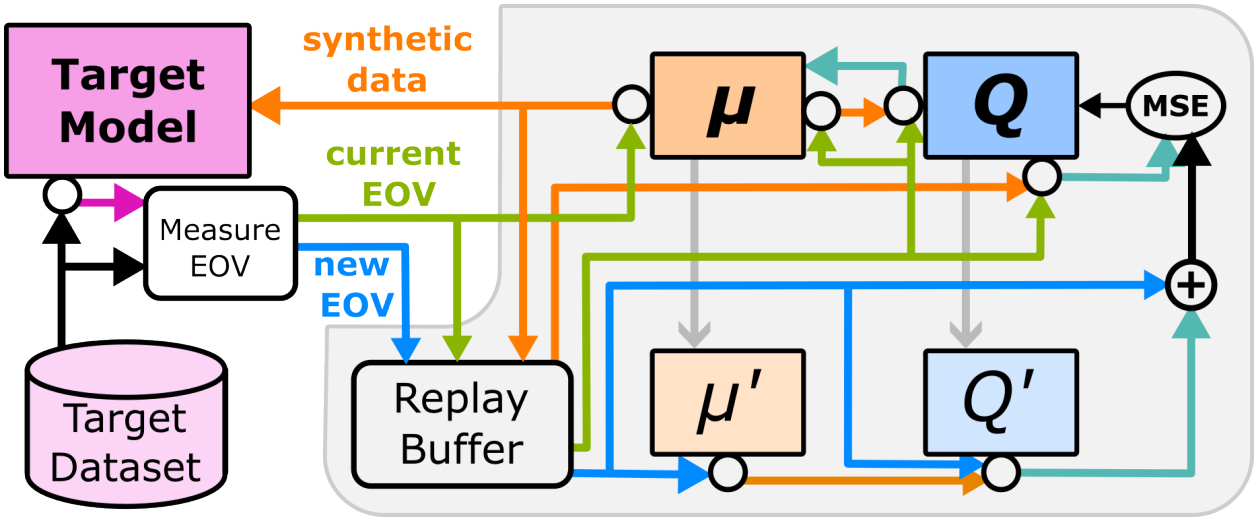

Figure 4.2: System Datapath Outline. As in Fig. 4.1, bubbles attached to each network denote a forward pass. Q' and μ' are the soft-copied secondary networks used in discounted future-state prediction. Note how the Target's dataset is used only by the EOV measuring system – if the system were self-sufficient, the dataset would be unnecessary.

The process is standard backpropagation training. First, the Agent's internal perceptrons

Algorithm 1 RL Debiasing Procedure (adapted from Lillicrap et al. [27])

procedure DEBIAS(T, E, S, N, ρ, σ, γ, τ)

Create critic networks $Q(b, \Delta b, a|\theta^Q)$ and Q', and actor networks $\mu(b, \Delta b|\theta^\mu)$ and μ'.

Randomly initialize weights θ^Q and θ^μ.

Copy weights to secondary networks $\theta^{Q'} \leftarrow \theta^Q, \theta^{\mu'} \leftarrow \theta^\mu$.

Initialize a ring buffer R of length ρ.

for episode = 1 to E **do**

Reset target model T to its initial state.

Clear ring buffer R.

Measure initial bias b_1 of T; $\Delta b_1 \leftarrow 0$

for step i=1 to S **do**

Generate synthetic datum $a_i = \mu(b_i, \Delta b_i|\theta^\mu) + \mathcal{N}(0, \sigma)$.

Train target model T on a_i.

Measure bias b'_i of T; $\Delta b'_i \leftarrow (b'_i - b_i)$.

Store $(b_i, \Delta b_i, a_i, b'_i, \Delta b'_i)$ in buffer R.

Sample a batch of N items $(b_n, \Delta b_n, a_n, b'_n, \Delta b'_n)$ from buffer R.

Set $y_n = b'_n + \gamma Q'(b'_n, \Delta b'_n, \mu'(b'_n, \Delta b'_n|\theta^{\mu'})|\theta^{Q'})$ for $n \in N$

Update critic by minimizing the loss $L = \frac{1}{N}\sum_n (y_n - Q(b_n, \Delta b_n, a_n|\theta^Q))^2$

Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_n \nabla_a Q(b_n, \Delta b_n, a|\theta^Q)|_{a=\mu(b_n, \Delta b_n)} \nabla_{\theta^\mu} \mu(b_n, \Delta b_n|\theta^\mu)|_{b_n, \Delta b_n}$$

Copy weights to the secondary networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

end for

end for

end procedure

are initialized (lines 2-4). We then enter into an episode loop ('episode' is the RL equivalent of 'epoch' and the two terms are used interchangeably) (line 6). Since each episode, in basic ML training, is meant to consist of a fresh pass over the training data, we reset the target model to its initial state (line 7) and clear the Agent's memory buffer (line 8).

Each episode is divided into a number of *steps* (line 10): in each step, the Agent receives an observation and chooses an action; that is, we measure the bias of the target model (line 9), and the Agent's Actor network uses this measurement to generate a synthetic datum (line 11). We then feed the datum into the Target model in training mode (line 12). By measuring the Target's bias again (line 13), we obtain both the reward for that datum and the next observation. The system can use any suitable bias metric; ours measures Equalized Odds, as discussed in Chapter 3. The prior observation, datum generated, reward thereof, and next observation together form a data tuple, which we store in the Agent's memory buffer (line 14).

The Actor-Critic Agent alone is entirely deterministic; what variation it experiences in its exploration is due entirely to its starting weights. This is not desirable – the Agent should be capable of stochastic exploration – so we implement such exploration by adding some noise to the actions. Once the Actor has chosen an action from an observation, we add a Gaussian noise sample to each feature of the synthetic datum (line 11). The result is stored as the action, as if the Actor had created it. The Gaussian distribution was used because it is simple and versatile; future work might investigate alternative noise distributions (Lillicrap et al. use an Ornstein-Uhlenbeck process). The distribution has a mean of zero and a hyperparametrized standard deviation.

At the end of each step, we perform one training pass for the Agent. In order to do so, we employ the Actor-Critic paradigm discussed above. We take a batch of observation/action/reward/next observation tuples from the Agent's memory (line 15), and train the Agent's Critic network to estimate the reward for actions given observations, that is, minimizing the Mean Squared Error between the Critic's predictions and the observed rewards using a library backpropagation optimizer (17). To give the Agent some capacity of

lookahead, as discussed above, we adjust the observed rewards by adding to them some proportion of the estimated reward for taking a presumed optimal action in the next state (line 16). The estimated reward is provided by the secondary Critic network, and the presumed optimal action is provided by the secondary Actor network.

We can now use the Critic network to train the Actor network (line 18). Lillicrap et al. [27] provide the key: their work shows that the Actor network's gradient can be approximated by taking the mean of the Critic's estimated reward for actions taken by the Actor, given observations. We therefore use the Actor to generate a batch of actions from the observations in the batch we drew from the Agent's memory. We then run the Critic on those observations and the freshly generated actions; this produces estimated rewards for each observation/action pair. The mean of these rewards gives us an approximate loss, which we can backpropagate to train the Actor network.

As the final part of each step we soft-copy the weights from the primary Actor and Critic networks to the secondary Actor and Critic networks (line 19); thus the latter are less responsive to rapid changes in the former induced by fluctuations in the training process.

4.3.1 Model hyperparameters

Algorithm 1 shows some of the available hyperparameters of the debiasing procedure. We will define them here (summarized in Table 4.1), and discuss choices for their values in the Methods section:

- Episode count E: The number of episodes (epochs); since each episode entails a reset of the target model, this hyperparameter controls the extent of the Agent's training. We found that four episodes were sufficient to obtain performant results; indeed, the EOV sometimes worsened after three episodes.

- Step count S: The number of steps per episode; how many actions the Agent generates, and how many synthetic training data the Target is subjected to. This hyperparameter was changed often during the search; our main breakthrough into performance

came from setting this value sufficiently high to allow the Agent time to do its work. Increasing it yet further yielded no improvements.

- Sample size N: At each step, this number of observation/action/reward/next observation tuples are sampled from the Agent's memory buffer to use in training. If there are insufficient tuples in the buffer, no training occurs.

- Memory size ρ: The number of tuples that the Agent's memory buffer can contain. The buffer is a FIFO queue: if a data tuple is added to the buffer when it is full, the oldest tuple is evicted. Neither this nor the Sample Size N were much changed during the search, as initial exploration suggested their impact was minimal. Further investigation is needed.

- Exploration process standard deviation σ: The Agent's exploration process is a Gaussian distribution with mean zero and this value for standard deviation. As the random values are added directly to the synthetic data, this value is in the Target model's unknowable feature-space. However, it is not unreasonable to assume that the Target model expects normalized training data, as normalizing training data is common practice, so the unknowable feature-space should be of roughly uniform scale across target domains. We discuss alternatives to Gaussian exploration in Chapter 7.

- Lookahead factor γ: The Agent's future prediction is multiplied by this factor before being added to the observed reward. Closer to zero means a more myopic agent; closer to one means the future prediction is given equal weight with the current observation. The value used was taken from Lillicrap et al.'s [27] work; experimenting with alternative values of γ did not improve performance.

- Softcopy factor τ: The strength of the copy from primary to secondary Actor/Critic networks. If closer to zero, the weights of the secondary networks will remain close to what they were; if closer to one, the weights will be overwritten by the weights of the

primary networks. As with the lookahead factor γ, the value was taken from Lillicrap et al.

Other hyperparameters not shown in the pseudocode are:

- Actor and Critic network specifications: The Agent's Actor and Critic are implemented as Multilayer Perceptrons. Their inputs and outputs are of known size (The Critic takes an observation and an action (two fixed-size tensors) and produces a reward (one real number), while the Actor takes an observation (one fixed-size tensor) and produces an action (one fixed-size tensor)), but the number and width of their hidden layers is not fixed, and must be chosen.

- Actor and Critic optimizers: Our implementation uses the Pytorch library, which has a variety of backpropagation optimizers available out of the box. The choice of algorithm affects convergence speed and which minimum the optimization process may find.

- Actor and Critic learning rates: Most of the aforementioned optimizers have a learning rate parameter, which is therefore a hyperparameter of the whole system. We chose to have our Critic learn ten times faster than our Actor, as the Critic is meant to model the observation-action-reward mapping, and this is more difficult to do the more volatile the Actor is.

4.3.2 Connecting the target model

Algorithm 1 shows that we require the target model to have three properties. Firstly, we must be able to train it: the Agent generates data which are fed into the target, so we must be able to provide the Target with said data and invoke its unknown learning procedure. Secondly, we need a means to restore the target to its initial configuration, in order to reset at the beginning of each episode. Lastly we need a means to evaluate the bias of the target. If the target exposed such a metric on its own, our system would be truly data-agnostic.

Table 4.1: System Hyperparameters

Name	Symbol	Type	Range	Value used
Episode count	E	Integer	$[1, \infty)$	4
Step count	S	Integer	$[1, \infty)$	100000
Sample size	N	Integer	$[1, \infty)$	1024
Memory size	ρ	Integer	$[1, \infty)$	4096
Exploration standard deviation	σ	Real	$[0.0, \infty)$	8.0
Lookahead factor	γ	Real	$[0.0, 1.0]$	0.99
Softcopy factor	τ	Real	$[0.0, 1.0]$	0.001
Actor learning rate	α^{μ}	Real	$[0.0, 1.0]$	10^{-7}
Actor hidden shape	$\lvert\theta^{\mu}\rvert$	MLP shape	N/A	$[400, 300]$
Critic learning rate	α^{Q}	Real	$[0.0, 1.0]$	10^{-6}
Critic hidden shape	$\lvert\theta^{Q}\rvert$	MLP shape	N/A	$[400, 300]$

However, this is not likely to be the case, so we measure the target's bias using a subset of the target's training data. By passing the training data into the target in evaluation mode, and applying one of the metrics discussed in Chapter 3 (Equalized Odds, in our case) on the results, we can measure the target's bias.

Chapter 5

EXPERIMENTAL METHOD

5.1 Tasks

To show that our new system meets its desired properties (laid out in Section 2.3.2), we perform the following experiments. The latter three desiderata (*modifies pre-trained models*, *no complexity increase*, and *single-pass post-factum training*) are met by the model's structure.

5.1.1 Target Model Architectures

To substantiate the claim that the system is *model-agnostic*, we trained four different ML models to act as biased targets: one Logistic Regression classifier and three Multilayer Perceptrons. The Perceptrons were given varying architectures: the 'normal' configuration had one hidden layer of 50 neurons, the 'wide' configuration had one hidden layer of 100 neurons, and the 'deep' configuration had two hidden layers of 50 neurons apiece. Each model had a Sigmoid output layer to perform binary classification, and each used a PyTorch library Adam optimizer with a learning rate of 10^{-5}. Each model was given between 10 and 30 epochs (depending on accuracy convergence) of pre-training on the target dataset before being fed into the debiasing system, with target accuracies being comparable to the baselines listed on the UC Irvine Machine Learning Repository [23].

5.1.2 Target Model Datasets

To demonstrate *data-agnosticism*, we trained our target models on three standard fairness-research demographic tabular datasets, in order to have state-of-the-art benchmarks for comparison:

- The Adult dataset [5], a set of cleaned US census data from 1996. It contains 48,842 data items with 14 features such as age, employment, and education. The associated classification task is to predict whether a given individual has an income above a certain threshold. The sensitive attribute used here was sex.

- The COMPAS dataset [3], a set of demographic data from defendants in Broward County, FL, used by ProPublica to show bias in Northpointe's COMPAS recidivism prediction algorithm. It contains 7214 data items (of which we used 6172, due to missing fields) with 13 features such as age, number of criminal priors, and charge degree. The task here is to predict whether a given defendant will reoffend in two years' time. For the sensitive attribute we used race (which we binarized as African-American vs. Non-African-American), as the sex attribute was highly skewed.

- The German Credit dataset [19], another set of demographic data from German loan applicants. It contains 1000 data items with 20 features apiece, such as credit history, employment status, and number of dependents. The task here classifies individuals into 'good' or 'bad' credit risks. The sensitive attribute used was sex, which had to be extracted from the dataset's original 'sex-and-status' feature; why these two were combined remains a mystery.

5.2 Benchmark Values

For each of the above datasets we compare our model's performance against state-of-the-art benchmarks, which employ various other debiasing techniques. For the Adult dataset, we have the work of Su et al. [36], who add a fairness term directly to their model's loss function, and Zhang et al. [49], who employ an adversarial strategy. For COMPAS, we have Su et al. again, as well as Zafar et al. [46], who add constraints to their model's decision boundary, and Wadsworth et al. [41], who use an adversarial strategy akin to Zhang et al. For the German Credit dataset, we have Biswas and Rajan [7], again with an adversarial strategy,

and Liao and Naghizadeh [26], who apply the direct optimization term of Agarwal et al. [1] to the German Credit task.

During our initial experiments in search of a system architecture, we experimented with an adversarial approach inspired by the work of Yang et al. [45]. Their approach (similar to the approach of Zhang et al., Wadsworth et al., and Biswas and Rajan) involves an adversarial model which attempts to recover the sensitive attribute of a datum given the prediction assigned it by the main model. The adversary's performance, inverted, is factored into the main model's loss function, the (empirically vindicated) assumption being that as the adversary lost the ability to predict the sensitive attribute, the main model's fairness would increase. That architecture was unsuitable for our system (as we cannot add terms to a loss function that may not even exist), but we implemented a prototype variant of it, and, as Yang et al. did not debias the three tasks we chose, we ran said prototype ourselves to obtain a guess of what they might have achieved if they had. These results compare very favourably to the other benchmarks, and we include them for reference.

Tables 5.1 - 5.3 show the relevant benchmarks for each task, with an initial biased measurement (from the initial state of our own basic-configuration target MLP, as trained to the UCI standards [23]) for comparison.

Table 5.1: Benchmarks for the Adult Dataset

	Su et al. (2023)[36]	Zhang et al. (2018)[49]	Custom Yang et al. (2023)[45]	Initial biased
min. EOV	0.071	0.011	0.050	0.293
accuracy	0.768	0.845	0.837	0.834

Table 5.2: Benchmarks for the COMPAS Dataset

	Su et al. (2023)[36]	Zafar et al. (2019)[46]	Wadsworth et al. (2018)[41]	Custom Yang et al. (2023)[45]	Initial biased
min. EOV	0.097	0.020	0.01	0.159	0.164
accuracy	0.573	0.654	0.70	0.648	0.576

Table 5.3: Benchmarks for the German Dataset

	Biswas and Rajan (2020)[7]	Liao and Naghizadeh (2023)[26]	Custom Yang et al. (2023)[45]	Initial biased
min. EOV	0.006	0.16	0.027	0.244
accuracy	0.729	0.74	0.760	0.718

5.3 Experimental Procedure

5.3.1 Hyperparameter optimization

To find performant values for the hyperparameters given in section 4.3.1, we trained a normal MLP target on a subset of the Adult dataset and employed a heuristic-guided Monte Carlo search process. This yielded a step count S of 100000, a sample size N of 1024, a memory size ρ of 4096, an exploration standard deviation σ of 8, a lookahead factor γ of 0.99, and a softcopy factor τ of 0.001. Actor and Critic MLPs were constructed from a 400-neuron hidden layer followed by a 300-neuron hidden layer, with Pytorch library Adam optimizers. The learning rate of the Actor was set at 10^{-7}, while the Critic's was set at 10^{-6}.

We do not believe these hyperparameters are optimal, as the possibility space is large and the interaction between hyperparameters is not well understood.

5.3.2 Trials

For each pairing of dataset and target architecture, we ran four trials: one trial with the Agent's learning process entirely disabled (for reasons we will discuss in Section 7.2), and three normal trials. In each trial, the debiasing system was given 4 episodes (hyperparameter E in section 4.3.1) to debias its target. Within the episode, the target's bias was measured using a training subset of the relevant dataset. Before the process and at the end of each episode, the target's bias was measured using a withheld testing subset. The target's classification accuracy was also measured. Chapter 6 shows the results so obtained.

Trials were run on two separate Mac laptops equipped with M1-Pro 10-core CPUs with 16 and 32 GB of RAM respectively; as the debiasing process is highly sequential, no advantage is gained from GPU parallelization; indeed, initial trials run on a server with 4 NVIDIA RTX 2080 Ti GPUs proved to be far slower than M1-Pro CPU execution on the laptops. Each episode took on average approximately 40 minutes to execute.

Chapter 6

RESULTS

6.1 Experimental Data

Figures 6.1 through 6.12 show the progress of the target model's bias as the debiasing Agent steps forward: the dim, noisy lines show the step-to-step Equalized Odds Violation measurement of target bias, computed using the training dataset, while the solid lines show EOV calculated at the end of each episode, using the withheld testing dataset.

Figures 6.13 through 6.15 show the summary of all results obtained, in addition to target model initial state and benchmarks for comparison. For each configuration, the three 'learning' trials were averaged to produce a single result. The figures show the minimum target EOV obtained, and the classification accuracy of the target at that minimal point, so as to demonstrate the fairness-accuracy tradeoff inherent in machine debiasing. Tables 6.1 through 6.3 present the same data, along with the standard deviation of the learning trials.

6.2 Summary of Experimental Results

These results suggest that it is indeed possible to automatically debias an ML model without regard to its architecture or its data, and, what is more, to obtain performance approaching that of state-of-the-art model- and data-gnostic debiasers.

For the Adult task (Fig. 6.13, Table 6.1), our system reduces the Equalized Odds Violation below that of the state-of-the-art benchmarks for the perceptron targets. The logistic target's EOV reduction is also close to SOTA techniques. The system loses a similar amount of target accuracy as the work of Su et al., but lags behind the better accuracies of Zhang et al. and Yang et al.

The German task results (Fig. 6.15, Table 6.3) are likewise promising. EOV reduction

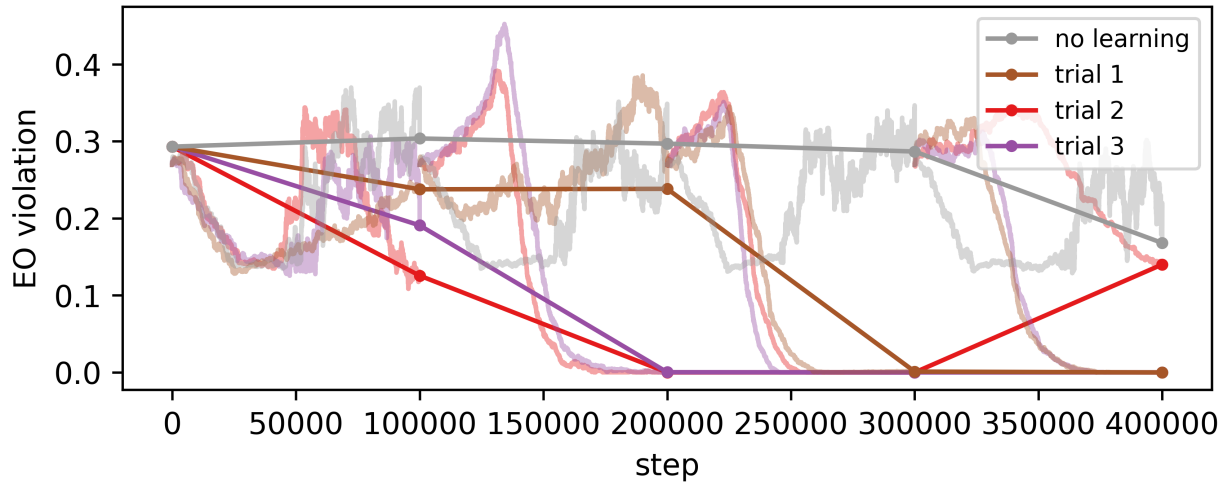

Figure 6.1: Training EOV on a basic MLP target, Adult task

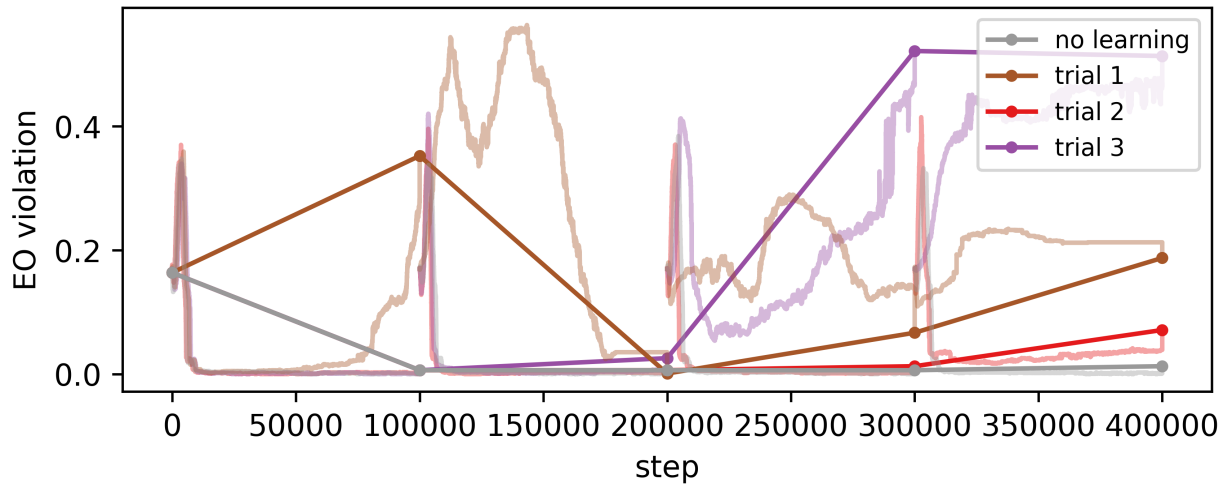

Figure 6.2: Training EOV on a basic MLP target, COMPAS task

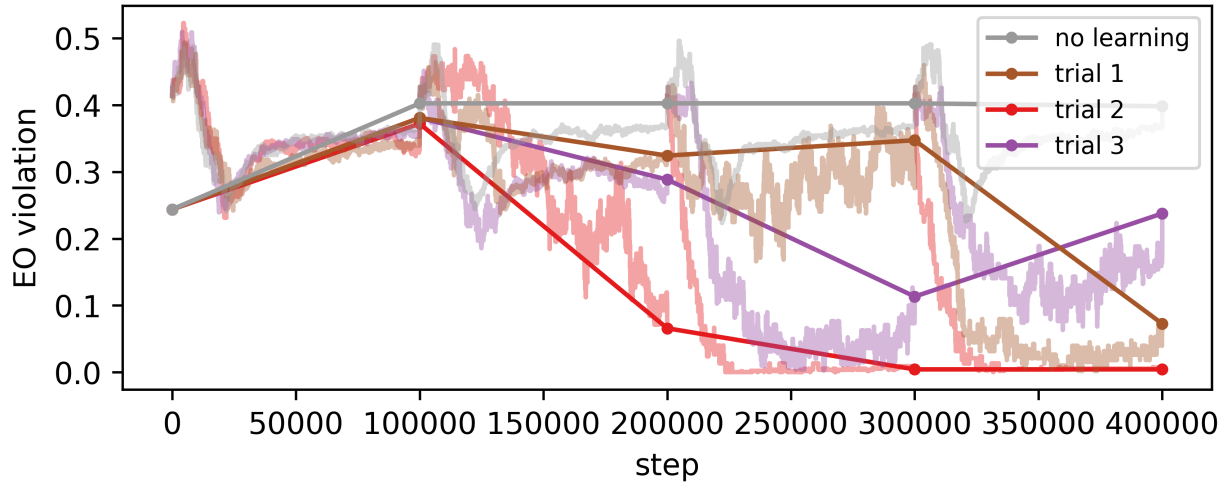

Figure 6.3: Training EOV on a basic MLP target, German Credit task

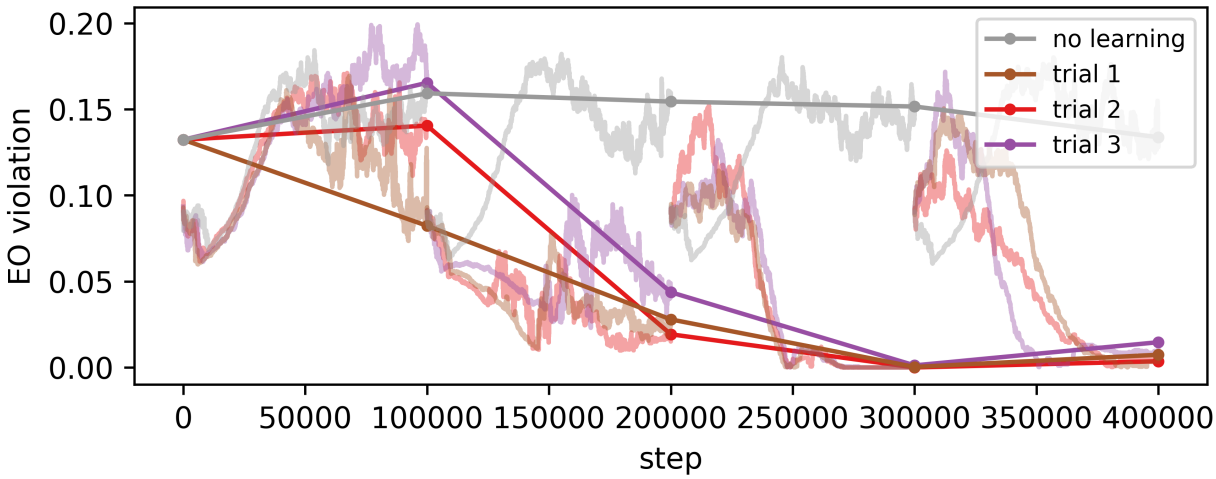

Figure 6.4: Training EOV on a wide MLP target, Adult task

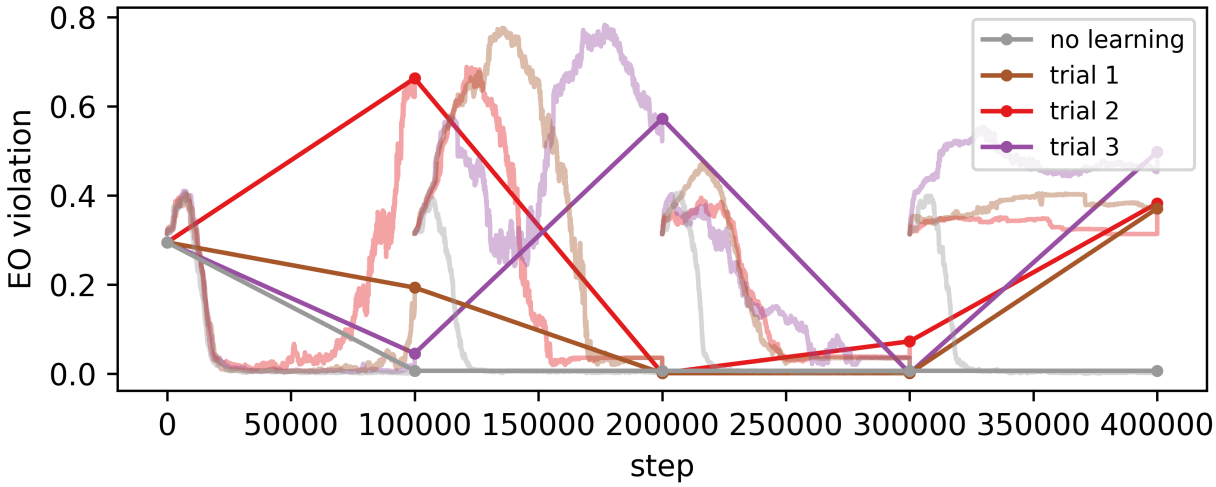

Figure 6.5: Training EOV on a wide MLP target, COMPAS task

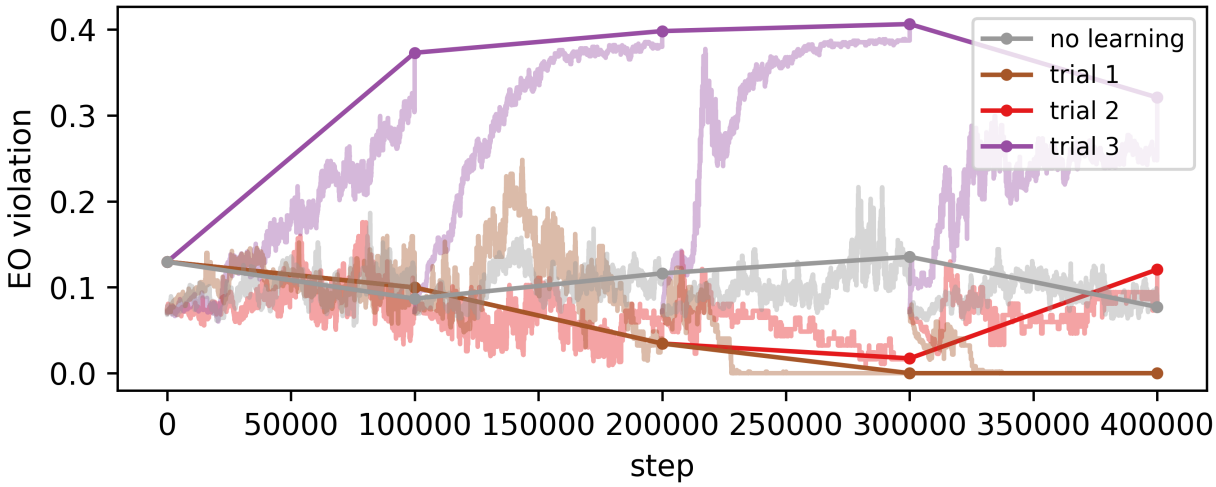

Figure 6.6: Training EOV on a wide MLP target, German Credit task

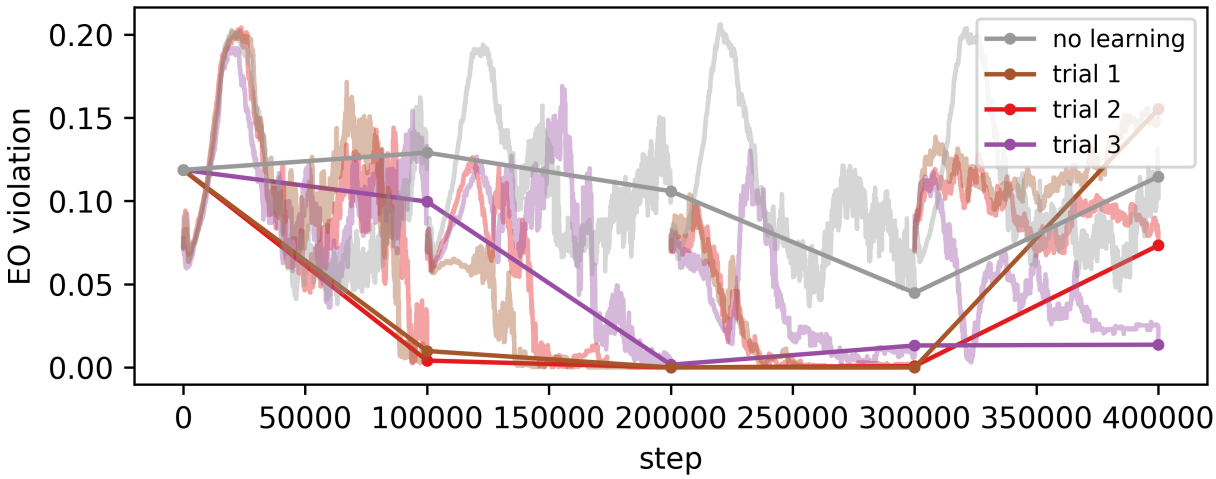

Figure 6.7: Training EOV on a deep MLP target, Adult task

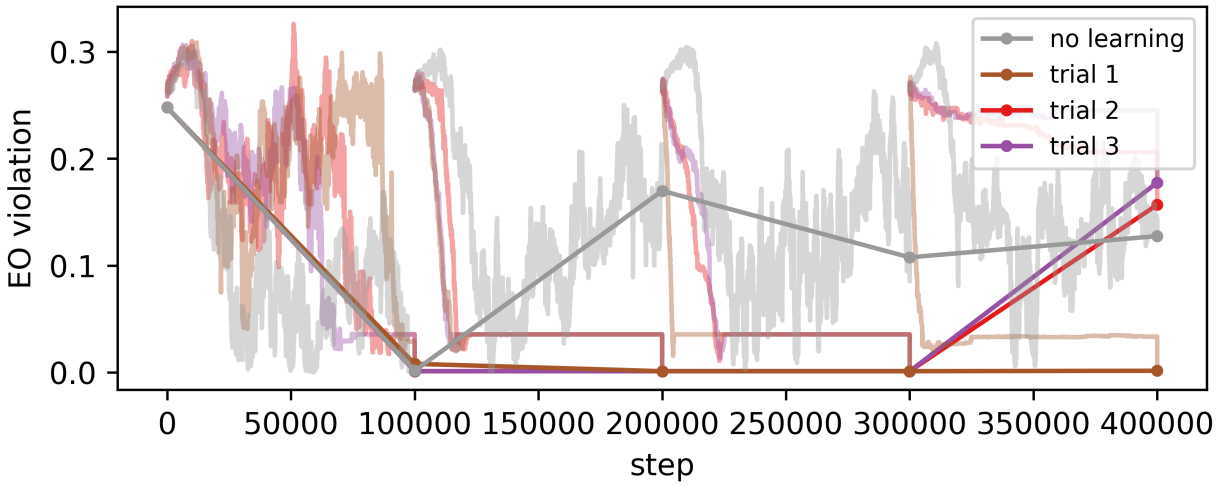

Figure 6.8: Training EOV on a deep MLP target, COMPAS task

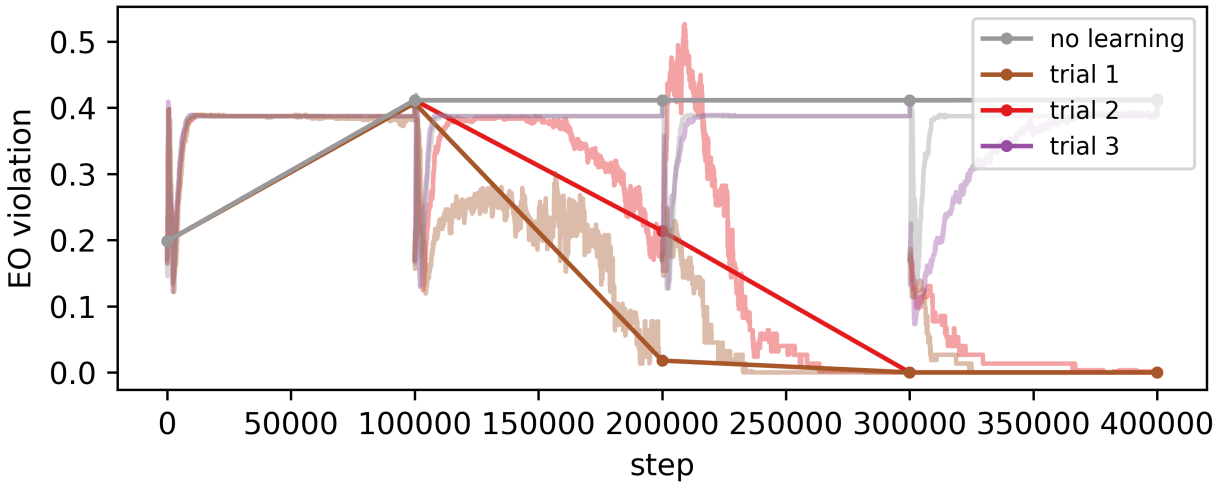

Figure 6.9: Training EOV on a deep MLP target, German Credit task

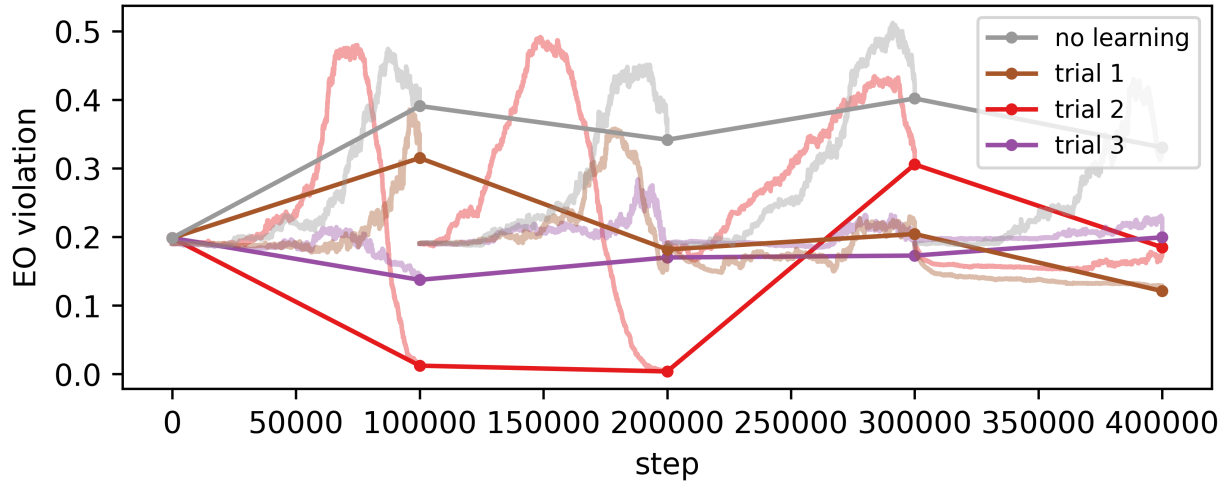

Figure 6.10: Training EOV on a Logistic Regression target, Adult task

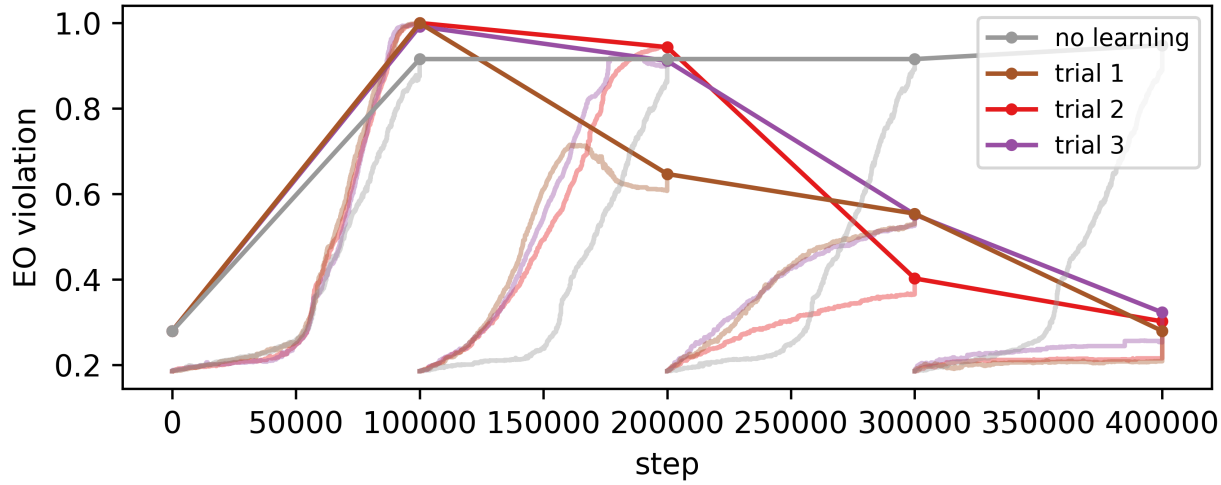

Figure 6.11: Training EOV on a Logistic Regression target, COMPAS task

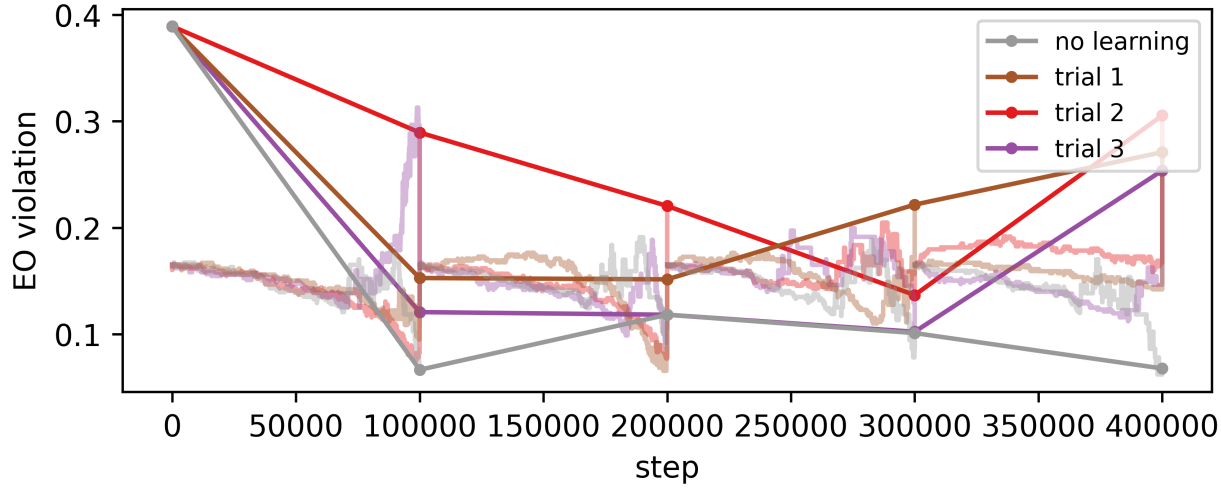

Figure 6.12: Training EOV on a Logistic Regression target, German Credit task

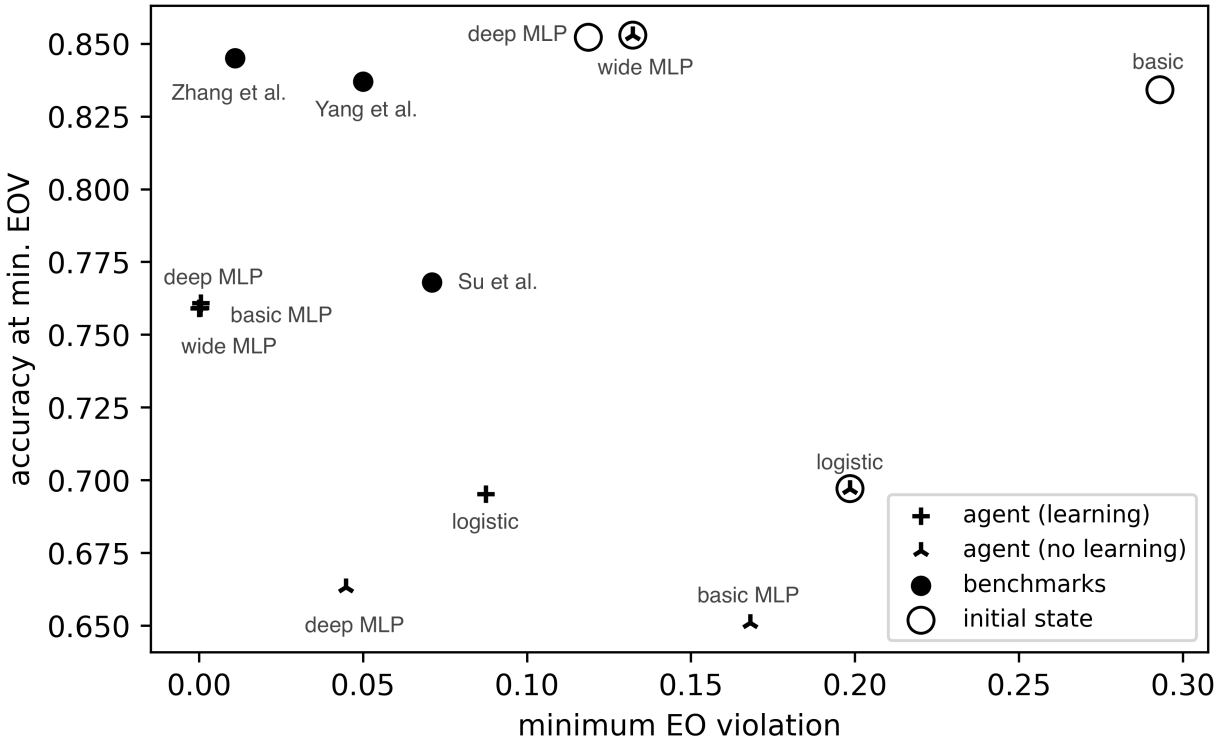

Figure 6.13: EOV/Accuracy tradeoff for Adult Task

Table 6.1: Results for the Adult Task. Learning Trial standard deviation given in parentheses.

	Basic MLP	Wide MLP	Deep MLP	Logistic Regression
initial EOV	0.293	0.132	0.119	0.198
min. EOV (learning)	0.000 (0.000)	0.000 (0.001)	0.001 (0.001)	0.087 (0.059)
min. EOV (no learning)	0.168	0.132	0.045	0.198
initial accuracy	0.834	0.853	0.852	0.697
accuracy (learning)	0.759 (0.000)	0.759 (0.000)	0.761 (0.002)	0.695 (0.064)
accuracy (no learning)	0.651	0.853	0.663	0.697

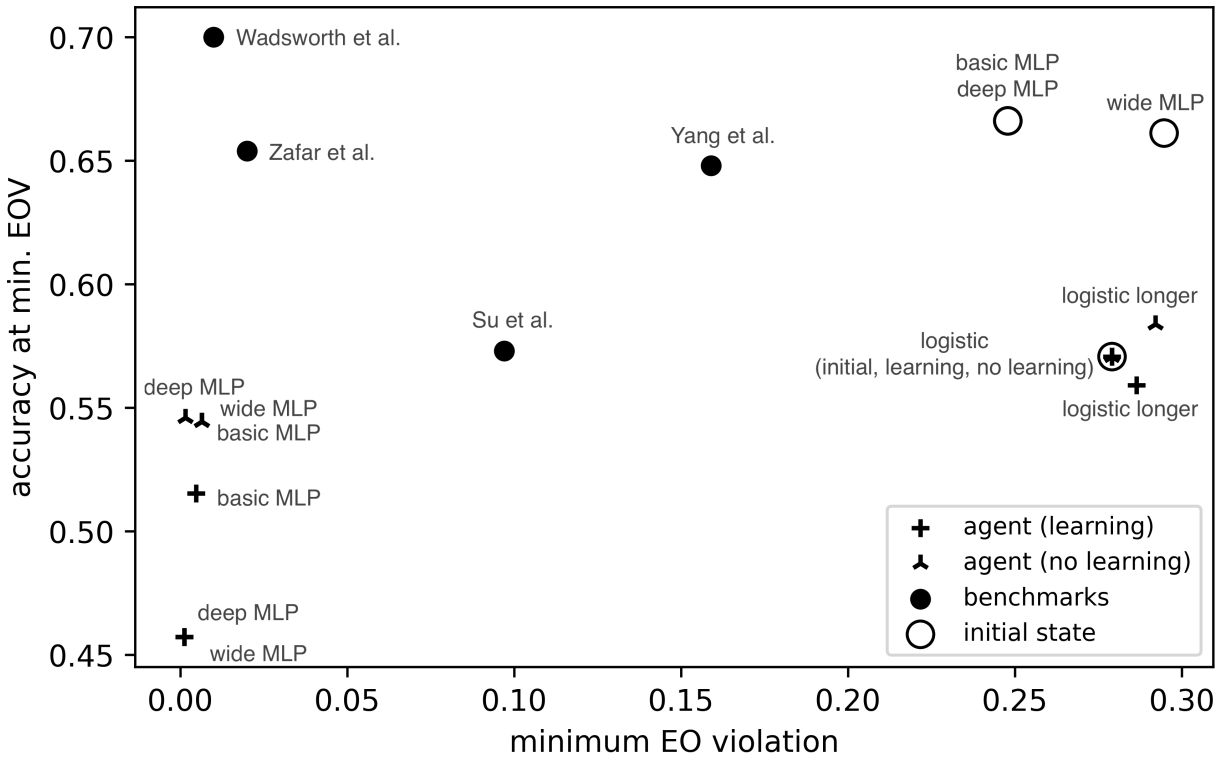

Figure 6.14: EOV/Accuracy tradeoff for COMPAS Task

Table 6.2: Results for the COMPAS Task. Learning Trial standard deviation given in parentheses.

	Basic MLP	Wide MLP	Deep MLP	Logistic Regression
initial EOV	0.164	0.295	0.248	0.279
min. EOV (learning)	0.005 (0.003)	0.001 (0.000)	0.001 (0.000)	0.279 (0.000)
min. EOV (no learning)	0.006	0.004	0.002	0.279
initial accuracy	0.576	0.661	0.666	0.571
accuracy (learning)	0.515 (0.041)	0.457 (0.000)	0.457 (0.000)	0.570 (0.000)
accuracy (no learning)	0.544	0.544	0.546	0.570

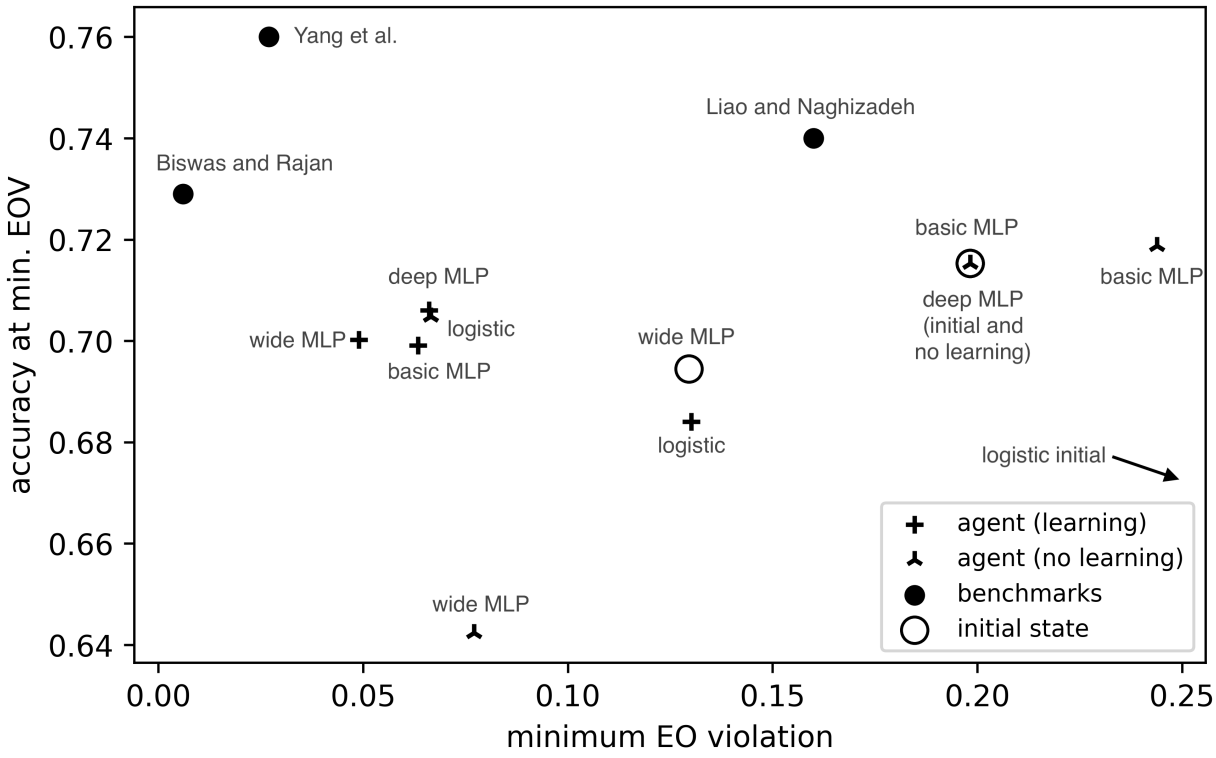

Figure 6.15: EOV/Accuracy tradeoff for German Task

Table 6.3: Results for the German Task. Learning Trial standard deviation given in parentheses.

	Basic MLP	Wide MLP	Deep MLP	Logistic Regression
initial EOV	0.244	0.130	0.198	0.389
min. EOV (learning)	0.063 (0.045)	0.049 (0.057)	0.066 (0.093)	0.130 (0.020)
min. EOV (no learning)	0.244	0.077	0.198	0.067
initial accuracy	0.719	0.694	0.715	0.573
accuracy (learning)	0.699 (0.007)	0.700 (0.004)	0.706 (0.006)	0.684 (0.009)
accuracy (no learning)	0.719	0.642	0.715	0.705

on all target architectures, including logistic regression, falls in the middle of the SOTA benchmarks. The accuracy loss is likewise good, with accuracy at minimum EOV only slightly below the initial state.

The COMPAS task results (Fig. 6.14, Table 6.2) are the weakest. The MLP target architectures experienced almost perfect EOV reduction, exceeding all SOTA benchmarks, but at the cost of unacceptable accuracy loss. For the logistic target, the system had little effect.

The training progress (Figures 6.1 through 6.12) reveal some considerations for practical application. Some of the learning trials of COMPAS basic (Fig. 6.2), COMPAS wide (Fig. 6.5), COMPAS deep (Fig. 6.8), Adult deep (Fig. 6.7), and German logistic (Fig. 6.12) exhibit a rebound, in which Equalized Odds violation strikes a minimum and then begins to increase. This observation suggests the utility of some best-result saving method or early-stopping criterion. In the German deep (Fig. 6.9) and German wide (Fig. 6.6) trials, the system displays a potential sensitivity to initial variation, in which certain trials increase EOV and appear to show no likelihood of decrease. Without further knowledge of what element of the initial condition causes this rapid divergence, one possible solution might be a component that restarts the debiasing if such an increase is detected.

Chapter 7

DISCUSSION

7.1 Limitations

7.1.1 Theoretical drawbacks

Section 4.3.2 briefly mentions one way in which our system falls short of its theoretical requirement of total data-agnosticity. We cannot evade the need for some means of measuring the target model's bias; since that bias depends on the model's classification of data with respect to said data's sensitive attributes, it cannot be computed without some access to that data. Our system used two subsets (training and testing) of each target's dataset to perform its bias calculations, and we cannot therefore claim that it is truly data-agnostic.

However, the algorithmic core of the system – the Reinforcement-Learning Agent – is data-agnostic; it requires only the data's shape and the bias measurements. If the bias measurements were provided by the target model through some opaque mechanism, the system would attain true data-agnosticity. Such a mechanism would of course also require data-access, but, if the measurement were done target-side, would address some of the reasons we wish the debiaser to be data-agnostic in the first place, namely, that the dataset might be secret, sensitive, or unwieldy.

Another theoretical problem concerns model-agnosticity. Our experiments were performed only on ML models that could accept further training data after being trained; this is not true of all ML models (for instance, Decision Trees). Our system might be able to debias these models also, through a baroque relaxation of the data-agnosticity requirement wherein the target is entirely retrained at each step with the synthetic debiasing data appended to its dataset. Such a procedure would probably be expensive.

7.1.2 Practical difficulties

One practical problem with our technique is that the target's classification accuracy drops below that of the SOTA benchmarks in all cases. Furthermore, in all cases but one (German Task, Logistic target) the learning Agent induced an accuracy drop below the initial target accuracy. These differences are in some cases slight, and in some cases major. They stem from a systemic problem know as *fairness-accuracy tradeoff* [6, 40, 33, 24]; that is, a classifier that must make allowances for fairness cannot reliably be as accurate as one that does not. Some debiasing techniques make this tradeoff an explicit, controllable value; others, like the work of Grgić-Hlača et al. [16], devise entire constrained optimization processes to eke out as much utility as possible. Our system makes no provision for accuracy at all, rather optimizing only for minimal Equalized Odds Violation. The targets remain accurate because their (black-box) optimizers are (presumably) optimizing for the target's own accuracy.

Another practical difficulty encountered was execution time. Our system is necessarily sequential, as the target's reaction to any given action depends on its state which is in turn directly modified by said action. The debiasing Agent can therefore only furnish the target with one synthetic datum at a time. ML libraries are written with parallelization as a goal, the understanding being that large batches of data can be drawn and passed through models simultaneously. As our system cannot do this, we cannot take advantage of GPU execution (indeed, bussing the data back and forth seems to incur a significant time penalty) and must run our system entirely on the CPU. The debiasing process is therefore slow, and this is likely to be exacerbated by large, complex target models, though we did not test any such.

7.2 Learning-Free Debiasing

During our hyperparameter search, we inadvertently disabled the learning component of our Agent (by disabling the target model's reset and choosing an episode size that meant that there were never enough observations in the memory buffer to perform a learning step). This meant that the synthetic data being fed to the target were being generated entirely by the

Agent's exploratory noise generator.

We were surprised to find that merely feeding noise to the target was sufficient to incur a reduction in EOV. For the Adult task on a deep MLP target, the noise alone had a better EOV reduction than the learning logistic-target trials, albeit with poor accuracy. The German task on wide MLP and logistic targets had EOV reductions comparable to the learning methods, and the logistic target trial proved as accurate as the learning systems. In the COMPAS task the pure-noise method performed even better, with the nigh-perfect EOV reduction of the learning trials and higher accuracy for all MLP targets. The most intriguing result, however, was the noise-only trial on a wide MLP target for the Adult task: while the EOV reduction was less than both SOTA benchmarks and the learning system, it was still appreciable, and the accuracy at minimum EOV was actually greater than both the initial condition and the best of the SOTA.

7.3 Future Work

The viability of the prototype opens many avenues for future investigation. In addition to expanded target- and dataset- compatibility testing, it would be worthwhile to determine if the Agent could be re-used: synthetic data created to debias one model trained on a dataset might also debias a different model trained on the same data.

Exploring alternative data modalities may also prove fruitful. The existing systems upon which we wished to improve and our own new system are confined to the domain of tabular data. There is no theoretical reason, however, why RL-based model- and data-agnostic debiasing could not be applied to other forms of data such as natural language processing or image modeling.

The prototype would also undoubtedly benefit from more concerted hyperparameter optimization (certain hyperparameters, such as the Agent's memory size and Actor and Critic internal architectures, were barely touched during the search process). Architectural overhauls may also prove a fruitful pursuit; as it stands, the Agent learns to generate synthetic data directly, which is then perturbed by its noise generator; what if the Agent were instead,

say, to learn per-feature parameters for a generative noise distribution?

Another mechanism to add might be deliberate consideration for the fairness-accuracy tradeoff as discussed above. Additionally, we made a strong assumption by creating experimental target models which optimized themselves for accuracy only. We may wish to apply our system to models that are optimizing for other desirable properties (say, privacy).

Potential alternative applications of our general system might also be considered. In a Federated Learning setting, for instance, the Agent might be inserted as an 'impostor client'. It would receive the aggregate model from the FL server, measure the aggregate's fairness using its own data supply (with accuracy limited, of course, by whether the client's data is a representative sample) and then generate an entirely synthetic client model, with ersatz weights, to be sent back to the server to reduce the server's bias. Such a system might have certain drawbacks to overcome: the feedback loop would perhaps be infeasibly slow (which opens another avenue for research, that of extracting information about the server's bias from as little feedback as possible), and the server might flag the synthetics as outliers to be culled (which leads into the field of *data poisoning*).

7.4 Implications of Our Work

A real-world application of our new system can expect to gain the practical advantages detailed in Section 2.3. Its being model-agnostic means we can apply our technique to any ML model to whom we can pass new training data, and whose bias we can measure, without needing to modify or observe the internal structure of said model. The system's data-agnosticity means we need not care about the dataset involved; we need not tailor our system to suit particular distributions, nor obtain access to data that may be private, secret, or proprietary. The post-training application of our technique means we can debias existing, pre-trained models; we do not have to intervene in the training process, nor trust the training party to have a vested interest in debiasing. Additionally, the system need only be applied once; it adds no complexity to the target, which ensures that the target model's prediction does not become slower or more expensive.

Our work also has a broader theoretical implication for the use of RL Agents in ML. We modified a measurable property (fairness) of a target model using only synthetic training data. It may be that our system could modify other properties of an ML model as well; all that is needful is a means to measure the property in question.

BIBLIOGRAPHY

[1] Alekh Agarwal, Alina Beygelzimer, Miroslav Dudík, John Langford, and Hanna Wallach. A Reductions Approach to Fair Classification, July 2018. arXiv:1803.02453 [cs].

[2] Ibrahim Alabdulmohsin and Mario Lucic. A Near-Optimal Algorithm for Debiasing Trained Machine Learning Models, August 2022. arXiv:2106.12887 [cs, stat].

[3] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. How We Analyzed the COMPAS Recidivism Algorithm. https://www.propublica.org/article/how-we-analyzed-the-compas-recidivism-algorithm.

[4] Solon Barocas, Moritz Hardt, and Arvind Narayanan. *Fairness and Machine Learning: Limitations and Opportunities*. fairmlbook.org, 2019.

[5] Barry Becker and Ronny Kohavi. Adult, 1996. Published: UCI Machine Learning Repository. https://archive.ics.uci.edu/dataset/2.

[6] Richard Berk, Hoda Heidari, Shahin Jabbari, Michael Kearns, and Aaron Roth. Fairness in Criminal Justice Risk Assessments: The State of the Art, May 2017. arXiv:1703.09207 [stat].

[7] Sumon Biswas and Hridesh Rajan. Do the Machine Learning Models on a Crowd Sourced Platform Exhibit Bias? An Empirical Study on Model Fairness. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 642–653, November 2020. arXiv:2005.12379 [cs, stat].

[8] Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word

Embeddings. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[9] Flavio Calmon, Dennis Wei, Bhanukiran Vinzamuri, Karthikeyan Natesan Ramamurthy, and Kush R Varshney. Optimized Pre-Processing for Discrimination Prevention. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[10] Sam Corbett-Davies, Emma Pierson, Avi Feller, Sharad Goel, and Aziz Huq. Algorithmic decision making and the cost of fairness, June 2017. arXiv:1701.08230 [cs, stat].

[11] Amit Datta, Michael Carl Tschantz, and Anupam Datta. Automated Experiments on Ad Privacy Settings: A Tale of Opacity, Choice, and Discrimination, March 2015. arXiv:1408.6491 [cs].

[12] Lucas Dixon, John Li, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. Measuring and Mitigating Unintended Bias in Text Classification. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 67–73, New Orleans LA USA, December 2018. ACM.

[13] André F.Cruz, Pedro Saleiro, Catarina Belém, Carlos Soares, and Pedro Bizarro. Promoting Fairness through Hyperparameter Optimization. In *2021 IEEE International Conference on Data Mining (ICDM)*, pages 1036–1041, December 2021. ISSN: 2374-8486.

[14] Benjamin Fish, Jeremy Kun, and Ádám D. Lelkes. A Confidence-Based Approach for Balancing Fairness and Accuracy, January 2016. arXiv:1601.05764 [cs].

[15] Sorelle A. Friedler, Carlos Scheidegger, and Suresh Venkatasubramanian. On the (im)possibility of fairness, September 2016. arXiv:1609.07236 [cs, stat].

[16] Nina Grgić-Hlača, Muhammad Bilal Zafar, Krishna P. Gummadi, and Adrian Weller. Beyond Distributive Fairness in Algorithmic Decision Making: Feature Selection for

Procedurally Fair Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), April 2018. Number: 1.

[17] Moritz Hardt, Eric Price, and Nathan Srebro. Equality of Opportunity in Supervised Learning, October 2016. arXiv:1610.02413 [cs].

[18] Andrés Domínguez Hernández and Vassilis Galanos. A toolkit of dilemmas: Beyond debiasing and fairness formulas for responsible AI/ML, March 2023. arXiv:2303.01930 [cs].

[19] Hans Hofmann. Statlog (German Credit Data), 1994. Published: UCI Machine Learning Repository. https://archive.ics.uci.edu/dataset/144.

[20] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement Learning: A Survey, April 1996. arXiv:cs/9605103.

[21] Faisal Kamiran and Toon Calders. Data preprocessing techniques for classification without discrimination. *Knowledge and Information Systems*, 33(1):1–33, October 2012.

[22] Faisal Kamiran, Asim Karim, and Xiangliang Zhang. Decision Theory for Discrimination-Aware Classification. In *2012 IEEE 12th International Conference on Data Mining*, pages 924–929, December 2012. ISSN: 2374-8486.

[23] Markelle Kelly, Rachel Longjohn, and Kolby Nottingham. The UCI Machine Learning Repository. https://archive.ics.uci.edu.

[24] Patrik Joslin Kenfack, Adil Mehmood Khan, S.M. Ahsan Kazmi, Rasheed Hussain, Alma Oracevic, and Asad Masood Khattak. Impact of Model Ensemble On the Fairness of Classifiers in Machine Learning. In *2021 International Conference on Applied Artificial Intelligence (ICAPAI)*, pages 1–6, May 2021.

[25] Yanhui Li, Linghan Meng, Lin Chen, Li Yu, Di Wu, Yuming Zhou, and Baowen Xu. Training Data Debugging for the Fairness of Machine Learning Software. In

2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE), pages 2215–2227, May 2022. ISSN: 1558-1225.

[26] Yiqiao Liao and Parinaz Naghizadeh. Social Bias Meets Data Bias: The Impacts of Labeling and Measurement Errors on Fairness Criteria, May 2023. arXiv:2206.00137 [cs].

[27] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, July 2019. arXiv:1509.02971 [cs, stat].

[28] Guan-Horng Liu. ghliu/pytorch-ddpg, April 2024. https://github.com/ghliu/pytorch-ddpg.

[29] Francesco Locatello, Gabriele Abbati, Tom Rainforth, Stefan Bauer, Bernhard Schölkopf, and Olivier Bachem. On the Fairness of Disentangled Representations, October 2019. arXiv:1905.13662 [cs, stat].

[30] Kristian Lum and James Johndrow. A statistical framework for fair predictive algorithms, October 2016. arXiv:1610.08077 [cs, stat].

[31] David Madras, Elliot Creager, Toniann Pitassi, and Richard Zemel. Learning Adversarially Fair and Transferable Representations, October 2018. arXiv:1802.06309 [cs, stat].

[32] Ninareh Mehrabi, Umang Gupta, Fred Morstatter, Greg Ver Steeg, and Aram Galstyan. Attributing Fair Decisions with Attention Interventions, September 2021. arXiv:2109.03952 [cs].

[33] Aditya Krishna Menon and Robert C. Williamson. The cost of fairness in classification, May 2017. arXiv:1705.09055 [cs].

[34] Giang Nguyen, Sumon Biswas, and Hridesh Rajan. Fix Fairness, Don't Ruin Accuracy: Performance Aware Fairness Repair using AutoML, June 2023. arXiv:2306.09297 [cs].

[35] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic Policy Gradient Algorithms. In *Proceedings of the 31st International Conference on Machine Learning*, pages 387–395. PMLR, January 2014. ISSN: 1938-7228.

[36] Cong Su, Guoxian Yu, Jun Wang, Hui Li, Qingzhong Li, and Han Yu. Multi-dimensional Fair Federated Learning, December 2023. arXiv:2312.05551 [cs].

[37] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999.

[38] Latanya Sweeney. Discrimination in Online Ad Delivery, January 2013.

[39] Zeyu Tang, Jiji Zhang, and Kun Zhang. What-Is and How-To for Fairness in Machine Learning: A Survey, Reflection, and Perspective. *ACM Computing Surveys*, May 2023. Just Accepted.

[40] Mike H.M. Teodorescu and Xinyu Yao. Machine Learning Fairness is Computationally Difficult and Algorithmically Unsatisfactorily Solved. In *2021 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–8, September 2021. ISSN: 2643-1971.

[41] Christina Wadsworth, Francesca Vera, and Chris Piech. Achieving Fairness through Adversarial Learning: an Application to Recidivism Prediction, June 2018. arXiv:1807.00199 [cs, stat].

[42] Dennis Wei, Karthikeyan Natesan Ramamurthy, and Flavio du Pin Calmon. Optimized Score Transformation for Consistent Fair Classification, October 2021. arXiv:1906.00066 [cs, math, stat].

[43] Blake Woodworth, Suriya Gunasekar, Mesrob I. Ohannessian, and Nathan Srebro. Learning Non-Discriminatory Predictors, November 2017. arXiv:1702.06081 [cs].

[44] Yisong Xiao, Aishan Liu, Tianlin Li, and Xianglong Liu. Latent Imitator: Generating Natural Individual Discriminatory Instances for Black-Box Fairness Testing, May 2023. arXiv:2305.11602 [cs].

[45] Jenny Yang, Andrew A. S. Soltan, David W. Eyre, Yang Yang, and David A. Clifton. An adversarial training framework for mitigating algorithmic biases in clinical machine learning. *npj Digital Medicine*, 6(1):1–10, March 2023. Number: 1 Publisher: Nature Publishing Group.

[46] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez-Rodriguez, and Krishna P. Gummadi. Fairness Constraints: A Flexible Approach for Fair Classification. *Journal of Machine Learning Research*, 20(75):1–42, 2019.

[47] Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. Learning Fair Representations. In *Proceedings of the 30th International Conference on Machine Learning*, pages 325–333. PMLR, May 2013. ISSN: 1938-7228.

[48] Huimin Zeng, Zhenrui Yue, Lanyu Shang, Yang Zhang, and Dong Wang. Boosting Demographic Fairness of Face Attribute Classifiers via Latent Adversarial Representations. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 1588–1593, December 2022.

[49] Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. Mitigating Unwanted Biases with Adversarial Learning, January 2018. arXiv:1801.07593 [cs].

[50] Jindi Zhang, Luning Wang, Dan Su, Yongxiang Huang, Caleb Chen Cao, and Lei Chen. Model Debiasing via Gradient-based Explanation on Representation, September 2023. arXiv:2305.12178 [cs].

Appendix A

SOURCE CODE

The source code for this thesis may be found at:

`https://github.com/pinkavat/rl_debiaser`