FairLib: A Unified Framework for Assessing and Improving Fairness

Xudong Han[1] **Aili Shen**[1,2*] **Yitong Li**[3] **Lea Frermann**[1]
Timothy Baldwin[1,4] **Trevor Cohn**[1]
[1]The University of Melbourne [2]Alexa AI, Amazon
[3]Huawei Technologies Co., Ltd. [4]MBZUAI
xudongh1@student.unimelb.edu.au ailishen@amazon.com
liyitong3@huawei.com {lfrermann,tbaldwin,t.cohn}@unimelb.edu.au

Abstract

This paper presents *FairLib*, an open-source Python library for assessing and improving model fairness. It provides a systematic framework for quickly accessing benchmark datasets, reproducing existing debiasing baseline models, developing new methods, evaluating models with different metrics, and visualizing their results. Its modularity and extensibility enable the framework to be used for diverse types of inputs, including natural language, images, and audio. It incorporates 14 debiasing methods, including pre-processing, at-training-time, and post-processing approaches. The built-in metrics cover the most commonly acknowledged fairness criteria, and can be further generalized and customized for fairness evaluation.[1]

1 Introduction

While neural methods have achieved great success, it has been shown that naively-trained models often learn spurious correlations with protected attributes like user demographics or socio-economic factors, leading to allocation harms, stereotyping, and other representation harms (Badjatiya et al., 2019; Zhao et al., 2018; Li et al., 2018; Díaz et al., 2018; Wang et al., 2019). As a result, there is a surge of interest in assessing and improving fairness.

Various bias evaluation metrics have been introduced in previous studies to gauge different types of biases. One common family of fairness assessment is *group fairness* which measures performance disparities across demographic groups. Different instantiations of group fairness have been proposed, including *demographic parity* (Feldman et al., 2015), where the positive prediction rate should be identical across groups (irrespective of the gold label), or *equal opportunity* (Hardt et al., 2016) where all groups should have an equal chance of false negative prediction (*equalized odds* extends the notion to include equal true positive rates). More recent work addressed disparities within classes and demographic groups (Shen et al., 2022b). While these approaches reflect the nature of fairness increasingly faithfully, they have been applied and evaluated inconsistently in previous work, which impedes systematic analysis and comparison of proposed approaches.

In terms of bias mitigation, diverse debiasing methods have been proposed, including at-training-time (Li et al., 2018; Elazar and Goldberg, 2018; Shen et al., 2022a), and pre- (Zhao et al., 2017; Wang et al., 2019) and post-processing approaches (Han et al., 2022a; Ravfogel et al., 2020). Although these methods have been proved effective for bias mitigation, it is challenging to reproduce results and compare methods because of inconsistencies in training strategy and model selection criteria, which demonstrably affect the results.

We present *FairLib*, a well-documented, open-source framework for assessing and improving fairness. *FairLib* implements a number of common debiasing approaches in a unified framework to facilitate reproducible and consistent evaluation, and provides interfaces for developing new debiasing methods. Moreover, a dataset interface supports adoption of both built-in and newly developed methods for new tasks and corpora. For better presentation, *FairLib* also provides utilities for result summarization and visualization.

FairLib is implemented in Python using PyTorch and is easy to use: it can be run from the command line, or imported as a package into other projects. To demonstrate its utility, we use *FairLib* to reproduce a battery of debiasing results from the recent NLP literature, and show that improved and systematic hyperparameter tuning leads to demonstrable improvements over the originally reported results. *FairLib* is released under Apache License 2.0 and

*This work was done when Aili Shen was at The University of Melbourne.

[1]Please check out the demo notebook and the demo video.

Proceedings of the The 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 60 - 71

is available on GitHub.[2] Detailed documentation and tutorials are available on *FairLib*'s website.[3]

2 Benchmark Datasets

In addition to evaluating bias wrt. a user group, we require datasets where each input instance is annotated with protected attributes (e.g., gender) and a target class label (e.g., sentiment). However, for a variety of reasons, only a small subset of datasets contains protected attribute labels, and annotating protected labels can be difficult.

To standardize fairness studies, *FairLib* provides APIs to access various publicly available fairness benchmark datasets, including: (1) text corpora for occupation classification (BIOS, De-Arteaga et al. (2019)), sentiment analysis (MOJI, Blodgett et al. (2016)), and part-of-speech tagging (TRUSTPILOT, Hovy (2015)); (2) structured data for the tasks of recidivism prediction (COMPAS, Larson et al. (2016)), and income prediction (ADULT, Kohavi et al. (1996)); and (3) image data to address colored handwritten digit recognition (COLOREDMNIST, Arjovsky et al. (2019)), objective classification (COCO, Zhao et al. (2017)), and event classification (IMSITU, Zhao et al. (2017)).[4]

3 Fairness Criteria

FairLib includes a variety of widely-used fairness evaluation metrics from the literature.

Representational Fairness: To evaluate whether sensitive information (such as demographics) is encoded in the representations of a trained model, previous work has proposed to estimate the *leakage* using an attacker (Elazar and Goldberg, 2018; Wang et al., 2019). Specifically, an attacker is trained to reverse-engineer protected attributes of inputs based on learned representations or the original inputs. *FairLib* provides flexible APIs to estimate information leakage at any representational level, based on different attackers (including linear and neural models).

Group Fairness: To evaluate whether model predictions are fair towards the protected attributes, Barocas et al. (2019) present formal definitions of three types of group fairness criteria, which capture different levels of (conditional) independence between the protected attribute g, the target variable y, and the model prediction $\hat{y}$. Table 1 summarizes the statistical fairness criteria and maps them to confusion-matrix-derived scores. The group fairness criteria evaluate the disparity of these scores across subgroups and classes.

Aggregation of subset performance metrics to a single figure of merit typically consists of two steps: (1) group-wise aggregation within each class, which reflects performance disparities across protected groups for each class; and (2) class-wise aggregation, to aggregate group-wise disparities for all classes (i.e., the vector from step 1) into a single number. The choice of aggregation function reflects different assumptions of fairness, and varies in previous work. Table 2 lists existing aggregation approaches which are built in to *FairLib*.[5]

4 Bias Mitigation

This section reviews the three primary types of debiasing methods, followed by Section 4.1, a summary of bias mitigation methods implemented in *FairLib*.

Pre-processing adjusts the training dataset to be balanced across protected groups before training, such that the input feature space is expected to be uncorrelated with the protected attributes. Typical approaches here adopt long-tail learning approaches for debiasing, such as resampling the training set such that the number of instances within each protected group is identical (Zhao et al., 2018; Wang et al., 2019; Han et al., 2022a).

At training time introduces constraints into the optimization process for model training. A popular method is adversarial training, which jointly trains: (i) a discriminator to recover protected attribute values; and (ii) the main model to correctly predict the target classes while at the same time preventing protected attributes from being correctly predicted (Wadsworth et al., 2018; Elazar and Goldberg, 2018; Li et al., 2018; Wang et al., 2019; Zhao and Gordon, 2019; Han et al., 2021).

Post-processing aims to adjust a trained classifier according to protected attributes, such that the final predictions are fair to different protected groups. For example, Ravfogel et al. (2020) iteratively project fixed text representations from a trained model to a null-space of protected attributes. Han et al. (2022a) adjust the predictions for each protected group by searching for the best prior for

[2] https://github.com/HanXudong/fairlib

[3] https://hanxudong.github.io/fairlib

[4] Check the *FairLib* website for a full list of built-in datasets.

[5] In Section 6.3, we further introduce a framework for generalized aggregation in *FairLib*.

Type	Main Idea	Metric (M)
Independence ($\hat{y} \perp g$)	Positive rate of each protected group is the same (*Demographic Parity*; Feldman et al. (2015))	$\frac{TP+FP}{TP+FP+TN+FN}$ (Positive Rate)
Separation ($\hat{y} \perp g \mid y$)	Acknowledges correlation between g and y (*Equalized Odds*; Hardt et al. (2016))	$\frac{TP}{TP+FN}$ (Recall or TPR) $\frac{FP}{FP+TN}$ (Fall-out or FPR)
Sufficiency ($y \perp g \mid \hat{y}$)	Predictions are calibrated for all groups (*Test Fairness*; Chouldechova (2017))	$\frac{TP}{TP+FP}$ (Precision) $\frac{TN}{TN+FN}$ (NPV)

Table 1: Built-in fairness evaluation metrics in *FairLib*.

Formulation	Reference
$\beta_c = \frac{1}{G}\sum_g \lvert M_{c,g} - \overline{M}_c \rvert$	Shen et al. (2022b)
$\beta_c = \frac{1}{G-1}\sum_g \lvert M_{c,g} - \overline{M}_c \rvert^2$	Lum et al. (2022)
$\beta_c = \max_g \lvert M_{c,g} - \overline{M}_c \rvert$	Yang et al. (2020)
$\beta_c = \min_g M_{c,g}$	Lahoti et al. (2020)
$\beta_c = \min_g \frac{M_{c,g}}{\overline{M}_c}$	Zafar et al. (2017)
$\beta_c = \max_g M_{c,g} - \min_g M_{c,g}$	Bird et al. (2020)
$\beta_c = \frac{\max_g M_{c,g}}{\min_g M_{c,g}}$	Feldman et al. (2015)
$\delta = \sqrt{\frac{1}{C}\sum_c \beta_c^2}$	Romanov et al. (2019)
$\delta = \frac{1}{C}\sum_c \beta_c$	Li et al. (2018)

Table 2: A subset of aggregation approaches for fairness evaluation from the literature that have are implemented in *FairLib*. C and G refer to the number of distinct classes and protected groups. $M_{c,g}$ is the evaluation results of class c and group g wrt. a particular evaluation metric M, such as TPR. β_c denotes the aggregation of group-wise disparities within class c, and following class-wise aggregation results in δ, which is the fairness score.

each group-specific component.

4.1 Implemented Methods

Table 3 lists 14 debiasing methods that are implemented in *FairLib*. It can be beneficial to employ different debiasing methods simultaneously (e.g., combine *pre-processing* and *training-time* methods (Wang et al., 2019; Han et al., 2022a)), which *FairLib* supports, and technically, every combination of these methods can be directly used without any further modifications.

5 Model Comparison

Typically, debiasing methods suffer from performance–fairness trade-offs, and no single method achieves both the best performance and fairness, making comparison between fairness methods difficult. In this section, we first introduce trade-off plots for model comparison, and then discuss model selection criteria that can be used

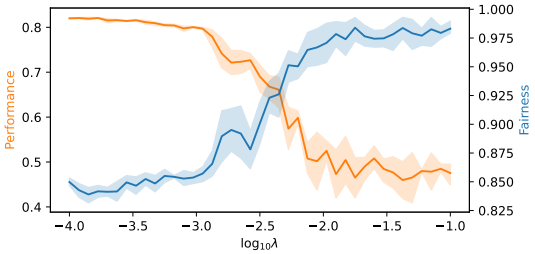


Figure 1: Tuning the tradeoff hyperparameter of FAIRSCL. Similar trade-offs can be obtained for other debiasing methods.

for reporting numerical results.

Performance–fairness Trade-off is a common way of comparing different debiasing methods without the requirement for model selection. Specifically, there is usually a trade-off hyperparameter for each debiasing method, which controls to what extent the model will sacrifice performance for better fairness, such as the number of iterations for null-space projection in INLP,[6] or the strength of the additional contrastive losses in FAIRSCL. Figure 1 shows a trade-off plot over different values of the trade-off hyperparameter of FAIRSCL for occupation classification, wherein we evaluate performance with accuracy, and use equal opportunity as the fairness criterion (see Section 8.1 for details).[7]

Instead of trade-offs wrt. different hyperparameter values, it can be more instructive to compute the maximum fairness that can be achieved by different models at a fixed performance level, and vice versa. Figure 2 shows an example of comparing the Pareto frontiers of INLP with FAIRSCL, where the results are obtained by varying the hyperparameters as illustrated in Figure 1. For a particular method, a Pareto optimal point corresponds to a model (i.e., a particular value of the trade-off hy-

[6] Cf., Table 3 for explanations of mentioned methods.

[7] Note that all figures and tables of results in this paper are direct outputs of *FairLib*.

Type	Model	Main Idea
Pre-	BD (Zhao et al., 2017)	Equalize the size of protected groups.
	CB (Wang et al., 2019)	Down-sample the majority protected group within each class.
	JB (Lahoti et al., 2020)	Jointly balance the Protected attributes and classes.
	BTEO (Han et al., 2022a)	Balance protected attributes within advantage classes.
At-	ADV (Li et al., 2018)	Prevent protected attributes from being identified by the discriminator.
	EADV (Elazar and Goldberg, 2018)	Employ multiple discriminators for adversarial training.
	DADV (Han et al., 2021)	Employ multiple discriminators with orthogonality regularization.
	AADV & ADADV (Han et al., 2022b)	Enable discriminators to use target labels as inputs during training.
	GATE (Han et al., 2022a)	Address protected factors with an augmented representation.
	FAIRBATCH (Roh et al., 2021)	Minimize CE loss gap though minibatch resampling.
	FAIRSCL (Shen et al., 2022a)	Adopt supervised contrastive learning for bias mitigation.
	EO_{CLA} (Shen et al., 2022b)	Minimize the CE loss gap within each target label by adjusting the loss.
Post-	INLP (Ravfogel et al., 2020)	Remove protected attributes through iterative null-space projection.
	GATE $^{\text{soft}}$ (Han et al., 2022a)	Adjust the prior for each group-specific component in GATE.

Table 3: Built-in methods for bias mitigation, which are grouped into three types: **Pre-**processing, **At** training time, and **Post-**processing.

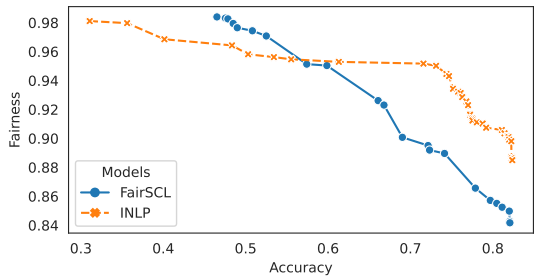


Figure 2: Pareto frontier curves derived from Figure 1.

perparameter) such that performance and fairness cannot be improved without causing a degradation in the other criterion.

Model Selection refers to the process of selecting the combination of hyperparameters that leads to best performance. In single-objective learning, model selection is based on a single metric, such as the loss on the dev set. In debiasing, however, both performance and fairness need to be considered for model selection, and a common method is *Constrained Selection*, which selects the best model given thresholds of the performance and fairness:

$$f^* = \arg\max_f q(f) \quad \text{s.t.} \quad \begin{matrix} \text{Perf}(f) > \text{h}_{\text{Perf}} \\ \text{Fair}(f) > \text{h}_{\text{Fair}} \end{matrix} \tag{1}$$

where f denotes a candidate model, $\text{Perf}(f)$ and $\text{Fair}(f)$ are the performance and fairness evaluation results for f, respectively, q is a real valued score function that maps the model f to a number, and h denotes corresponding thresholds. For instance, using $q(f) = \text{Fair}(f)$ results in the selection of the fairest candidate model.

Instead of measuring performance and fairness separately, one can explicitly measure their trade-off as the distance from a particular model f to the optimal point[8] (DTO, Han et al. (2021)):

$$\text{DTO}(f) = \sqrt{(1 - \text{Perf}(f))^2 + (1 - \text{Fair}(f))^2}\,,$$

which originates from the multi-objective optimization literature (Marler and Arora, 2004). Lower is better, with an optimal value of 0. Note that DTO should be minimized in Equation (1).

$\text{DTO}(f)$ is the default q function in *FairLib*. *FairLib* also supports the definition of customized cues, such as $\text{Perf}(f)$, $\text{Fair}(f)$, and $\text{DTO}(f)$. Given the flexibility of *FairLib*, most selection criteria in previous work can be reproduced, such as: (1) the maximum performance (Lahoti et al., 2020; Roh et al., 2021), which is based on a particular utility metric, such as accuracy and F-measures; (2) constrained selection (Han et al., 2021; Subramanian et al., 2021); and (3) minimising DTO (Han et al., 2022b; Shen et al., 2022b).

6 *FairLib* Design and Architecture

Here, we describe the four modules of *FairLib*, namely data, model, evaluation, and analysis.

6.1 Data Module

The data module manages inputs, target labels, and protected attributes for model training and evaluation. To enable different pre-processing debiasing methods in supporting any types of inputs, the `BaseDataset` class is implemented for sampling and weight calculation based on the distribution of classes and protected attributes. `Dataset` classes inherit functionality from `BaseDataset` with an additional property for loading different types of inputs.

[8]The optimum point is assumed to be a model that achieves 1 performance and 1 fairness. See Appendix B for details.

Specifically, *FairLib* includes `Dataset` classes for vector, matrix, and sequential inputs, to support structural, image, and text inputs. Once inputs are loaded by `Dataset`, pre-processing debiasing methods are automatically applied.

6.2 Model Module

This is the core module of *FairLib*, which implements the *At-training-time* and *Post-processing* debiasing methods described in Section 4.1 and Table 3. The methods can be applied to instances of the `BaseModel` class. One built-in child class of `BaseModel` is an `MLP` classifier for structural inputs, which can be fully integrated with HuggingFace's `transformers` library.[9] Specifically, the `MLP` can be used as the task-specific output layer, on top of the backbone networks from `transformers` (e.g. BERT (Devlin et al., 2019)), to handle a wide variety of inputs and tasks.

FairLib supports the combination of different bias mitigation methods with thousands of pre-trained models across classification tasks and data types, including text, image, and audio modalities.

6.3 Evaluation Module

This module implements the fairness metrics described in Section 3, and several performance measures. Performance measures are based on the classification evaluation metrics implemented in scikit-learn (Buitinck et al., 2013), including Accuracy, F-score, and ROC AUC. However, no established fairness evaluation suite exists. Noting that the calculation of existing fairness metrics is always based on confusion matrices, *FairLib* includes an `Evaluator` class which can: (1) calculate any confusion-matrix based fairness metrics; and (2) conduct group-wise and class-wise aggregations as specified by users.

6.4 Analysis Module

This module provides utilities for model comparison as introduced in Section 5, with the three main functions of: (1) conducting post-hoc early-stopping and model selection in parallel as introduced in Section 5;[10] (2) organizing the results as a Pandas DataFrame (pandas development team, 2020), which can be used to create plots and LaTeX tables;[11] and (3) creating interactive plots, covering different comparison settings such as Figures 2 and 4.[12]

7 Usage

In this section, we demonstrate the basic use of *FairLib*. For further details, see the online interactive demos for examples of adding customized models, datasets, and metrics.

The following command shows an example for training and evaluating a STANDARD model:

```
python fairlib --dataset Bios_gender
↪ --emb_size 768 --num_classes 28
↪ --encoder_architecture BERT
```

where the task dataset, the number of distinct classes, the encoder architecture, and the dimension of embeddings extracted from the corresponding encoder need to be specified. The above case trains a BERT classifer over the BIOS dataset, where there are 28 professions.

In order to apply built-in debiasing methods, additional options for debiasing can be added to the command-line to realise combinations of methods:

```
python fairlib --dataset Bios_gender
↪ --emb_size 768 --num_classes 28
↪ --encoder_architecture BERT --BT
↪  Resampling  --BTObj EO
↪ --adv_debiasing --INLP
```

The above example employs BTEO (*Pre-*), ADV (*At-*), and INLP (*Post-*) at the same time for a BERT classifer debiasing over the BIOS dataset.

FairLib can also be imported as a Python library; see Appendix D for more examples.

8 Benchmark Experiments

To evaluate *FairLib*, we conduct extensive experiments to compare models implemented in *FairLib* with their original reported results over two benchmark datasets. In Appendix A, we provide more experimental details.

8.1 Settings

We conduct experiments over two NLP classification tasks — sentiment analysis (MOJI) and biography classification (BIOS) — using the same dataset splits as previous work (Elazar and Goldberg, 2018; Ravfogel et al., 2020; Han et al., 2021; Shen et al., 2022a; Han et al., 2022a).

[9] https://github.com/huggingface/transformers

[10] Multi-processing is supported through the `joblib` library.

[11] All results are stored for later analysis, and are publicly available here.

[12] See here for more examples.

Method	Moji				Bios			
	Performance↑	Fairness↑	DTO ↓	Δ ↑	Performance↑	Fairness↑	DTO ↓	Δ ↑
STANDARD	72.30 ± 0.46	61.19 ± 0.44	47.68	0.56	82.25 ± 0.24	85.11 ± 0.81	23.17	0.69
BTEO	75.39 ± 0.14	87.75 ± 0.38	27.49	6.25	83.83 ± 0.25	90.54 ± 0.91	18.73	4.04
ADV	75.64 ± 0.73	89.33 ± 0.56	26.59	7.37	81.66 ± 0.22	90.74 ± 0.77	20.54	2.23
DADV	75.55 ± 0.41	90.40 ± 0.12	26.27	5.23	81.85 ± 0.19	90.64 ± 0.48	20.42	2.29
ADADV	75.02 ± 0.69	90.87 ± 0.17	26.60	0.00	81.91 ± 0.34	88.96 ± 0.59	21.19	0.00
FAIRBATCH	75.06 ± 0.60	90.55 ± 0.50	26.67	1.99	82.24 ± 0.13	89.50 ± 1.25	20.63	0.51
FAIRSCL	75.73 ± 0.34	87.82 ± 0.43	27.15	0.73	82.06 ± 0.16	84.27 ± 0.83	23.86	1.01
EO_{CLA}	75.28 ± 0.50	89.23 ± 0.79	26.97	0.25	81.78 ± 0.27	88.87 ± 0.94	21.35	1.13
INLP	73.34	85.60	30.30	15.90	82.30	88.62	21.04	9.21

Table 4: Evaluation results ± standard deviation (%) on the test set of MOJI and BIOS tasks, averaged over 5 runs with different random seeds. Δ: the DTO improvement of *FairLib* to the reported results in previous work. See Appendix A.2 for dataset statistics.

Following Han et al. (2022a), we report the overall Accuracy as the performance, and the Equal Opportunity as the fairness criterion, calculated based on the Recall gap across all protected groups.

8.2 Experimental Results

Table 4 summarizes the results produced by *FairLib*. Compared with previous work, STANDARD, ADADV, FAIRSCL and EO_{CLA} achieve similar results to the original paper. In contrast, the re-implemented BTEO, ADV, DADV, FAIRBATCH, and INLP outperform the results reported in their original paper due to the better-designed hyperparameter tuning and model selection.[13]

9 Related Work

Several toolkits have been developed for learning fair AI models (Bellamy et al., 2018; Saleiro et al., 2018; Bird et al., 2020). We discuss the two most closely-related frameworks.

The most related work to *FairLib* is AI Fairness 360 (*AIF360*), which is the first toolkit to bring together bias detection and mitigation (Bellamy et al., 2018). Like *FairLib*, *AIF360* supports a variety of fairness criteria and debiasing methods, and is designed to be extensible. The biggest difference over *FairLib* is that *AIF360* is closely tied to scikit-learn, and does not support other ML frameworks such as PyTorch. This not only limits the applicability of *AIF360* to NLP and CV tasks where neural model architectures are now de rigeur, but also implies a lack of GPU support. Moreover, *AIF360* only provides fundamental analysis features, such as comparing debiasing wrt. a single evaluation metric, while the analysis module of *FairLib* has richer features for model comparison, for example, selecting Pareto-models and interactive visualization.

The second closely-related library is *FairLearn* (Bird et al., 2020), which is also targeted at assessing and improving fairness for both classification and regression tasks. However, similar to *AIF360*, *FairLearn* is mainly developed for scikit-learn, meaning complex CV and NLP tasks are not supported. Additionally, *FairLearn* currently only supports four debiasing algorithms,[14] as opposed to the 14 methods supported in *FairLib*, providing fuller coverage of different debiasing methods.

In summary, *FairLib* complements existing fairness libraries by: (1) implementing a broad range of competitive debiasing approaches, with a specific focus on debiasing neural architectures which underlie many CV and NLP tasks; and (2) comprehensive tools for interactive model comparison to help users explore the effects of different debiasing approaches.

10 Conclusion

In this paper, we present *FairLib*, a new open-source Python library and framework for measuring and improving fairness, which implements a wide range of fairness metrics and 14 debiasing approaches. With better-designed hyperparameter tuning and model selection, the reproduced models in *FairLib* outperform the results reported in the original work. *FairLib* also has remarkable flexibility and extensibility, such that new models, debiasing methods, and datasets can be easily developed and evaluated.

[13] We provide further details of hyperparameter tuning in an online document.

[14] https://fairlearn.org/main/user_guide/mitigation.html

Acknowledgements

We thank the anonymous reviewers for their helpful feedback and suggestions. This work was funded by the Australian Research Council, Discovery grant DP200102519. This research was undertaken using the LIEF HPC-GPGPU Facility hosted at the University of Melbourne. This Facility was established with the assistance of LIEF Grant LE170100200.

Ethical Considerations

This work provides an unified framework for measuring and improving fairness. Although *FairLib* assumes access to training datasets with protected attributes, this is the same data assumption made by all debiasing methods. To avoid harm and be trustworthy, we only use attributes that have been publicly disclosed or the user has self-identified, or toy datasets. All data in this study is publicly available and used under strict ethical guidelines.

References

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. 2019. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*.

Pinkesh Badjatiya, Manish Gupta, and Vasudeva Varma. 2019. Stereotypical bias removal for hate speech detection task using knowledge-based generalizations. In *The World Wide Web Conference*, pages 49–59.

Solon Barocas, Moritz Hardt, and Arvind Narayanan. 2019. *Fairness and Machine Learning*. http://www.fairmlbook.org.

Rachel KE Bellamy, Kuntal Dey, Michael Hind, Samuel C Hoffman, Stephanie Houde, Kalapriya Kannan, Pranay Lohia, Jacquelyn Martino, Sameep Mehta, Aleksandra Mojsilovic, et al. 2018. Ai fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias. *arXiv preprint arXiv:1810.01943*.

Sarah Bird, Miro Dudík, Richard Edgar, Brandon Horn, Roman Lutz, Vanessa Milan, Mehrnoosh Sameki, Hanna Wallach, and Kathleen Walker. 2020. Fairlearn: A toolkit for assessing and improving fairness in ai. *Microsoft, Tech. Rep. MSR-TR-2020-32*.

Su Lin Blodgett, Lisa Green, and Brendan O'Connor. 2016. Demographic dialectal variation in social media: A case study of African-American English. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1119–1130.

Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. 2013. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122.

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

Maria De-Arteaga, Alexey Romanov, Hanna Wallach, Jennifer Chayes, Christian Borgs, Alexandra Chouldechova, Sahin Geyik, Krishnaram Kenthapadi, and Adam Tauman Kalai. 2019. Bias in bios: A case study of semantic representation bias in a high-stakes setting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 120–128.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Mark Díaz, Isaac Johnson, Amanda Lazar, Anne Marie Piper, and Darren Gergle. 2018. Addressing age-related bias in sentiment analysis. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–14.

Yanai Elazar and Yoav Goldberg. 2018. Adversarial removal of demographic attributes from text data. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 11–21.

Michael Feldman, Sorelle A Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. 2015. Certifying and removing disparate impact. In *proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 259–268.

Xudong Han, Timothy Baldwin, and Trevor Cohn. 2021. Diverse adversaries for mitigating bias in training. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2760–2765.

Xudong Han, Timothy Baldwin, and Trevor Cohn. 2022a. Balancing out bias: Achieving fairness through training reweighting. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP 2022)*. To appear.

Xudong Han, Timothy Baldwin, and Trevor Cohn. 2022b. Towards equal opportunity fairness

through adversarial learning. *arXiv preprint arXiv:2203.06317*.

Moritz Hardt, Eric Price, and Nati Srebro. 2016. Equality of opportunity in supervised learning. *Advances in Neural Information Processing Systems*, 29:3315–3323.

Dirk Hovy. 2015. Demographic factors improve classification performance. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 752–762.

Ron Kohavi et al. 1996. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Kdd*, volume 96, pages 202–207.

Preethi Lahoti, Alex Beutel, Jilin Chen, Kang Lee, Flavien Prost, Nithum Thain, Xuezhi Wang, and Ed Chi. 2020. Fairness without demographics through adversarially reweighted learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 728–740.

Jeff Larson, Surya Mattu, Lauren Kirchner, and Julia Angwin. 2016. How we analyzed the compas recidivism algorithm.

Yitong Li, Timothy Baldwin, and Trevor Cohn. 2018. Towards robust and privacy-preserving text representations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 25–30.

Kristian Lum, Yunfeng Zhang, and Amanda Bower. 2022. De-biasing "bias" measurement. In *2022 ACM Conference on Fairness, Accountability, and Transparency (FAccT '22)*, Seoul, Republic of Korea. ACM.

R Timothy Marler and Jasbir S Arora. 2004. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6):369–395.

The pandas development team. 2020. pandas-dev/pandas: Pandas. Zenodo.

Shauli Ravfogel, Yanai Elazar, Hila Gonen, Michael Twiton, and Yoav Goldberg. 2020. Null it out: Guarding protected attributes by iterative nullspace projection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7237–7256.

Yuji Roh, Kangwook Lee, Steven Euijong Whang, and Changho Suh. 2021. Fairbatch: Batch selection for model fairness. In *Proceedings of the 9th International Conference on Learning Representations*.

Alexey Romanov, Maria De-Arteaga, Hanna Wallach, Jennifer Chayes, Christian Borgs, Alexandra Chouldechova, Sahin Geyik, Krishnaram Kenthapadi, Anna Rumshisky, and Adam Kalai. 2019. What's in a name? reducing bias in bios without access to protected attributes. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4187–4195.

Pedro Saleiro, Benedict Kuester, Loren Hinkson, Jesse London, Abby Stevens, Ari Anisfeld, Kit T Rodolfa, and Rayid Ghani. 2018. Aequitas: A bias and fairness audit toolkit. *arXiv preprint arXiv:1811.05577*.

Aili Shen, Xudong Han, Trevor Cohn, Timothy Baldwin, and Lea Frermann. 2022a. Does representational fairness imply empirical fairness? In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing*. Association for Computational Linguistics. To appear.

Aili Shen, Xudong Han, Trevor Cohn, Timothy Baldwin, and Lea Frermann. 2022b. Optimising equal opportunity fairness in model training. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4073–4084, Seattle, United States. Association for Computational Linguistics.

Shivashankar Subramanian, Xudong Han, Timothy Baldwin, Trevor Cohn, and Lea Frermann. 2021. Evaluating debiasing techniques for intersectional biases. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2492–2498, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Christina Wadsworth, Francesca Vera, and Chris Piech. 2018. Achieving fairness through adversarial learning: an application to recidivism prediction. *FAT/ML Workshop*.

Tianlu Wang, Jieyu Zhao, Mark Yatskar, Kai-Wei Chang, and Vicente Ordonez. 2019. Balanced datasets are not enough: Estimating and mitigating gender bias in deep image representations. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5310–5319.

Forest Yang, Mouhamadou Cisse, and Sanmi Koyejo. 2020. Fairness with overlapping groups; a probabilistic perspective. *Advances in neural information processing systems*, 33:4067–4078.

Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rogriguez, and Krishna P Gummadi. 2017. Fairness constraints: Mechanisms for fair classification. In *Artificial Intelligence and Statistics*, pages 962–970. PMLR.

Han Zhao and Geoff Gordon. 2019. Inherent tradeoffs in learning fair representations. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2017. Men also like shopping: Reducing gender bias amplification using corpus-level constraints. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2979–2989.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018. Gender bias in coreference resolution: Evaluation and debiasing methods. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 15–20.

A Experimental Details

A.1 Datasets

MOJI: This sentiment analysis dataset was collected by Blodgett et al. (2016), and contains tweets that are either African American English (AAE)-like or Standard American English (SAE)-like. Each tweet is annotated with a binary 'race' label (based on language use: either AAE or SAE), and a binary sentiment score determined by (redacted) emoji contained in it.

BIOS: The second task is biography classification (De-Arteaga et al., 2019), where biographies were scraped from the web, and annotated for binary gender and 28 classes of profession.

A.2 Results Statistics

For each hyperparameter combination, we repeat experiments 5 time with different random seeds drawn from a discrete uniform distribution. The mean values and standard deviation are calculated based on the 5 runs. Due to the fact that INLP is a *post-processing* approach and its results with respect a given number of iterations are highly affected by the random seed, we only report results for 1 run. One way of getting statistics of INLP is selecting the trade-off hyperparameter of INLP for each random seed, however, this may not be a fair comparison with other methods as fixed hyperparameters have been used.

B Model Comparison

Figure 3 illustrates the key ideas of model comparison.

C Experimental Results

Trade-off plots for the selected methods are shown in Figure 4. Over the MOJI dataset (Figure 4a), it can be seen that almost all methods lead to similar results, with a fairness score less than 0.9, except for INLP, which is substantially worse than the other methods. As increasing the values of each model's trade-off hyperparameter (i.e., achieving better fairness at the cost of performance), ADADV outperforms other methods.

The trade-off plot for BIOS is quite different to MOJI: (1) INLP becomes a reasonable choice; (2) FAIRSCL does not work well over this dataset, consistent with the original paper; (3) BTEO is the only method that achieves better performance than the STANDARD model while increasing fairness; (4) EO_{CLA} could be the best choice as it achieves much better fairness than others at a comparable performance level.

D Further Usage

In this section, we demonstrate how to use *FairLib*. Users can run existing models or add their own models, datasets, and metrics as needed.

D.1 Basics

FairLib also support YAML configuration files with training options:

```
python fairlib --conf_file opt.yaml
```

which is useful for reproducing experimental results, as *FairLib* saves the YAML file for each run.

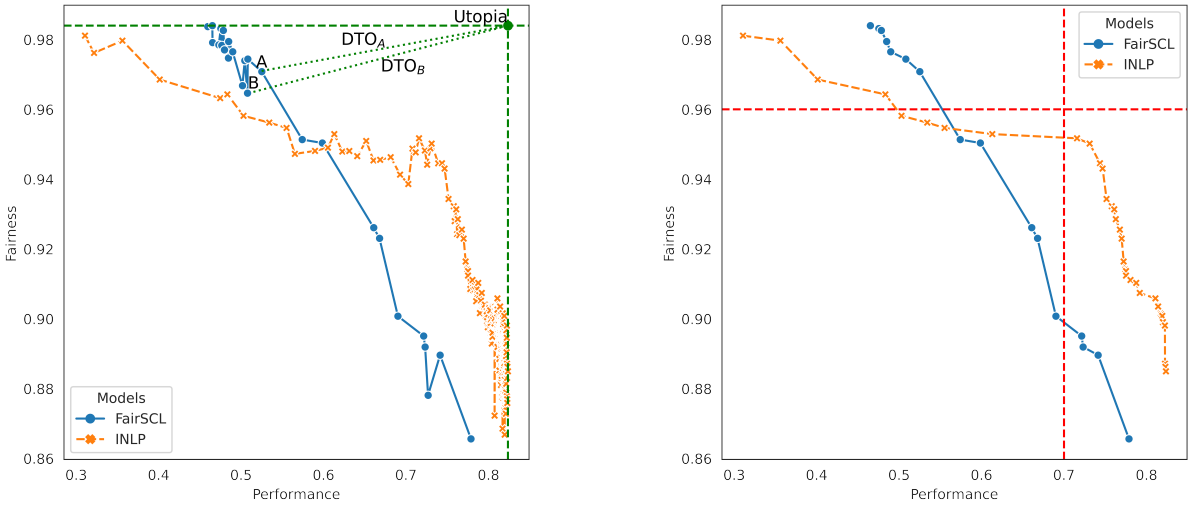
```
from fairlib.base_options import options
from fairlib import networks

config_file = 'opt.yaml'
# Get options
state =
↪  options.get_state(conf_file=config_file)

# Init the model
model = networks.get_main_model(state)

# Training with debiasing
model.train_self()
```

Checkpoints, evaluation results, outputs, and the configuration file are saved to the default or a specified directory.

D.2 Performing Analysis

As introduced in Section 6.4, the first step to analyze a trained model is selecting the best epoch. Here we provide an example for retrieving experimental results for FAIRSCL, and selecting the best epoch-checkpoint:

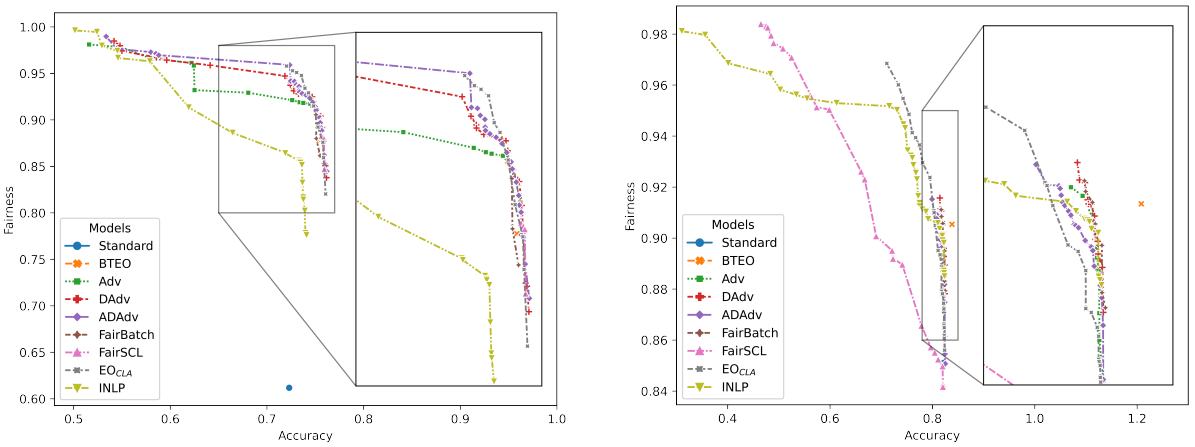
```
from fairlib.load_results import
↪  model_selection_parallel

FairSCL_df = model_selection(
  model_id= "FSCL",
  GAP_metric_name = "TPR_GAP",
  Performance_metric_name = "accuracy",
  selection_criterion = "DTO",
  n_jobs=20,
  index_column_names = ["fcl_lambda_y",
  ↪  "fcl_lambda_g"],
  save_path = "FairSCL_df.pkl",)
```



(a) Trade-off

(b) Pareto Trade-off

Figure 3: performance–fairness trade-offs of FAIRSCL (blue points) and INLP (orange crosses) over the BIOS dataset. The vertical and horizontal red dashed line in Figure 3b are examples of constrained model selection wrt. a performance threshold of 0.7 and fairness threshold of 0.96. Figure 3a also provides an example for DTO. The green dashed vertical and horizontal lines denote the best performance and fairness, respectively, and their intersection point is the Utopia point. The length of green dotted lines from A and B to the Utopia point are the DTO for candidate models A and B, respectively.



(a) MOJI

(b) BIOS

Figure 4: Performance–fairness trade-offs of selected models over the MOJI and BIOS datasets.

where the fairness metric is TPR GAP (corresponding to *Equal Opportunity* fairness); the performance is measured with Accuracy score; the best epoch is selected based on DTO; and the tuned trade-off hyperparameters are used as the index. `n_jobs` is an optional argument for multiprocessing, and the resulting DataFrame will be saved to the specified directory.

Assuming `Bios_gender_results` is a Python dictionary of retrieved experimental results from the first step, indexed by the corresponding method name, we provide the following function for model comparison:

```
from fairlib.tables_and_figures import
↪  final_results_df

Bios_results = {
  "INLP":INLP_df,
  "FairSCL":FairSCL_df,}

Bios_gender_main_results =
↪  final_results_df(
  results_dict = Bios_results,
  pareto = True,
  selection_criterion = "DTO",
```

```
  return_dev = True,)
```

where model selection is performed based on DTO. Each method has one selected model in the resulting DataFrame, which can then be used to create tables.

If visualization is desired, users can disable model selection by setting `selection_criterion = None`, in which case all Pareto frontier points will be returned.

D.3 Customized Datasets

A custom dataset class must implement the `load_data` function. Take a look at this sample implementation; the split is stored in a directory `self.data_dir`. The `args.data_dir` is either loaded from the arguments `-data_dir` or from the default value. `split` has three possible string values, `"train"`, `"dev"`, `"test"`, indicating the split that will be loaded.

Then the `load_data` function must assign the value of `self.X` as inputs, `self.y` as target labels, and `self.protected_label` as information for debiasing, such as gender, age, and race.

```
from fairlib.dataloaders.utils import
↪  BaseDataset

class SampleDataset(BaseDataset):
  def load_data(self):
    # Load data from pickle file
    filename = self.split+"df.pkl"
    _Path = self.args.data_dir /
    ↪  filename
    data = pd.read_pickle(_Path)

    # Save loaded data
    self.X = data["X"]
    self.y = data["y"]
    self.protected_label =
    ↪  data["protected_label"]
```

As a child class of `BaseDataset`, *Pre-processing* related operations will be automatically applied to the `SampleDataset`.

D.4 Customized Models

Recall that our current `MLP` implementation (Section 6.2) can be used as a classification head for different backbone models, and the new model will support all built-in debiasing methods.

Take a look at the following example: we use BERT as the feature extractor, and then use the extracted features as the input to the MLP classifier to make predictions.

We only need to define three functions: (1) `__init__`, which is used to initialize the model with pretrained BERT parameters, MLP classifier, and optimizer; (2) `forward`, which is the same as before, where we extract sentence representations then use the MLP to make predictions; and (3) `hidden`, which is used to get hidden representations for adversarial training.

```
from transformers import BertModel
from fairlib.networks.utils import
↪  BaseModel

class BERTClassifier(BaseModel):
  model_name = 'bert-base-cased'

  def __init__(self, args):
    super(BERTClassifier,
    ↪  self).__init__()
    self.args = args

    # Load pretrained model parameters.
    self.bert =
      BertModel.from_pretrained(
        self.model_name)

    # Init the classification head
    self.classifier = MLP(args)

    # Init optimizer, criterion, etc.
    self.init_for_training()

  def forward(self, input_data,
  ↪  group_label = None):
    # Extract representations
    bert_output =
    ↪  self.bert(input_data)[1]

    # Make predictions
    return self.classifier(bert_output,
    ↪  group_label)

  def hidden(self, input_data,
  ↪  group_label = None):
    # Extract representations
    bert_output =
    ↪  self.bert(input_data)[1]

    return self.classifier.hidden(
      bert_output, group_label)
```