Diverse Adversaries for Mitigating Bias in Training

Xudong Han **Timothy Baldwin** **Trevor Cohn**
School of Computing and Information Systems
The University of Melbourne
Victoria 3010, Australia
xudongh1@student.unimelb.edu.au
{tbaldwin,tcohn}@unimelb.edu.au

Abstract

Adversarial learning can learn fairer and less biased models of language than standard methods. However, current adversarial techniques only partially mitigate model bias, added to which their training procedures are often unstable. In this paper, we propose a novel approach to adversarial learning based on the use of multiple *diverse* discriminators, whereby discriminators are encouraged to learn orthogonal hidden representations from one another. Experimental results show that our method substantially improves over standard adversarial removal methods, in terms of reducing bias and the stability of training.

1 Introduction

While NLP models have achieved great successes, results can depend on spurious correlations with protected attributes of the authors of a given text, such as gender, age, or race. Including protected attributes in models can lead to problems such as leakage of personally-identifying information of the author (Li et al., 2018a), and unfair models, i.e., models which do not perform equally well for different sub-classes of user. This kind of unfairness has been shown to exist in many different tasks, including part-of-speech tagging (Hovy and Søgaard, 2015) and sentiment analysis (Kiritchenko and Mohammad, 2018).

One approach to diminishing the influence of protected attributes is to use adversarial methods, where an encoder attempts to prevent a discriminator from identifying the protected attributes in a given task (Li et al., 2018a). Specifically, an adversarial network is made up of an attacker and encoder, where the attacker detects protected information in the representation of the encoder, and the optimization of the encoder incorporates two parts: (1) minimizing the main loss, and (2) maximizing the attacker loss (i.e., preventing protected attributes from being detected by the attacker). Preventing protected attributes from being detected tends to result in fairer models, as protected attributes will more likely be independent rather than confounding variables. Although this method leads to demonstrably less biased models, there are still limitations, most notably that significant protected information still remains in the model's encodings and prediction outputs (Wang et al., 2019; Elazar and Goldberg, 2018).

Many different approaches have been proposed to strengthen the attacker, including: increasing the discriminator hidden dimensionality; assigning different weights to the adversarial component during training; using an ensemble of adversaries with different initializations; and reinitializing the adversarial weights every t epochs (Elazar and Goldberg, 2018). Of these, the ensemble method has been shown to perform best, but independently-trained attackers can generally still detect private information after adversarial removal.

In this paper, we adopt adversarial debiasing approaches and present a novel way of strengthening the adversarial component via orthogonality constraints (Salzmann et al., 2010). Over a sentiment analysis dataset with racial labels of the document authors, we show our method to result in both more accurate and fairer models, with privacy leakage close to the lower-bound.[1]

2 Methodology

Formally, given an input x_i annotated with main task label y_i and protected attribute label g_i, a main task model M is trained to predict $\hat{y}_i = M(x_i)$, and an adversary, aka "discriminator", A is trained to predict $\hat{g}_i = A(h_{M,i})$ from M's last hidden layer representation $h_{M,i}$. In this paper, we treat

[1] Source code available at https://github.com/HanXudong/Diverse_Adversaries_for_Mitigating_Bias_in_Training

Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics, pages 2760–2765
April 19 - 23, 2021.

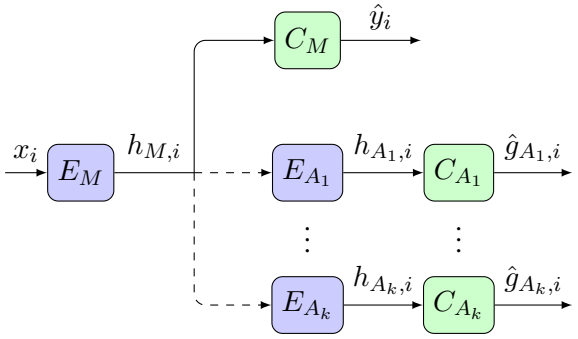

Figure 1: Ensemble adversarial method. Dashed lines denote gradient reversal in adversarial learning. The k sub-discriminators A_i are independently initialized. Given a single input x_i, the main task encoder computes a hidden representation $h_{M,i}$, which is used as the input to the main model output layer and sub-discriminators. From the k-th sub-discriminator, the estimated protected attribute label is $\hat{g}_{A_k,i} = C_{A_k}(E_{A_k}(h_{M,i}))$.

a neural network classifier as a combination of two connected parts: (1) an encoder E, and (2) a linear classifier C. For example, in the main task model M, the encoder E_M is used to compute the hidden representation $h_{M,i}$ from an input x_i, i.e., $h_{M,i} = E_M(x_i)$, and the decoder is used to make a prediction, $\hat{y}_i = C_M(h_{M,i})$. Similarly, for a discriminator, $\hat{g}_i = A(h_{M,i}) = C_A(E_A(h_{M,i}))$.

2.1 Adversarial Learning

Following the setup of Li et al. (2018a) and Elazar and Goldberg (2018) the optimisation objective for our standard adversarial training is:

$$\min_M \max_A \mathcal{X}(y, \hat{y}_M) - \lambda_{\text{adv}} \mathcal{X}(g, \hat{g}_A),$$

where $\mathcal{X}$ is cross entropy loss, and λ_{adv} is the trade-off hyperparameter. Solving this minimax optimization problem encourages the main task model hidden representation h_M to be informative to C_M and uninformative to A. Following Ganin and Lempitsky (2015), the above can be trained using stochastic gradient optimization with a gradient reversal layer for $\mathcal{X}(g, \hat{g}_A)$.

2.2 Differentiated Adversarial Ensemble

Inspired by the ensemble adversarial method (Elazar and Goldberg, 2018) and domain separation networks (Bousmalis et al., 2016), we present *differentiated adversarial ensemble*, a novel means of strengthening the adversarial component. Figure 1 shows a typical ensemble architecture where k sub-discriminators are included in the adversarial component, leading to an averaged adversarial regularisation term:

$$-\frac{\lambda_{\text{adv}}}{k} \sum_{j \in \{1,\ldots,k\}} \mathcal{X}(g, \hat{g}_{A_j}).$$

One problem associated with this ensemble architecture is that it cannot ensure that different sub-discriminators focus on different aspects of the representation. Indeed, experiments have shown that sub-discriminator ensembles can weaken the adversarial component (Elazar and Goldberg, 2018). To address this problem, we further introduce a difference loss (Bousmalis et al., 2016) to encourage the adversarial encoders to encode different aspects of the private information. As can be seen in Figure 1, $h_{A_k,i}$ denotes the output from the k-th sub-discriminator encoder given a hidden representation $h_{M,i}$, i.e., $h_{A_k,i} = E_{A_k}(h_{M,i})$.

The difference loss encourages orthogonality between the encoding representations of each pair of sub-discriminators:

$$\mathcal{L}_{\text{diff}} = \lambda_{\text{diff}} \sum_{i,j \in \{1,\ldots,k\}} \left\| {h_{A_i}}^\intercal h_{A_j} \right\|_F^2 \mathbb{1}(i \neq j),$$

where $\|\cdot\|_F^2$ is the squared Frobenius norm.

Intuitively, sub-discriminator encoders must learn different ways of identifying protected information given the same input embeddings, resulting in less biased models than the standard ensemble-based adversarial method. According to Bousmalis et al. (2016), the difference loss has the additional advantage of also being minimized when hidden representations shrink to zero. Therefore, instead of minimizing the difference loss by learning rotated hidden representations (i.e., the same model), this method biases adversaries to have representations that are a) orthogonal, and b) low magnitude; the degree to which is given by weight decay of the optimization function.

2.3 INLP

We include Iterative Null-space Projection ("INLP": Ravfogel et al. (2020)) as a baseline method for mitigating bias in trained models, in addition to standard and ensemble adversarial methods. In INLP, a linear discriminator (A_{linear}) of the protected attribute is iteratively trained from pre-computed fixed hidden representations (i.e., h_M) to project them onto the linear discriminator's

Model	Accuracy↑	TPR Gap↓	TNR Gap↓	Leakage@$\mathbf{h}$↓	Leakage@$\hat{y}$↓
Random	50.00±0.00	0.00±0.00	0.00±0.00	—	—
Fixed Encoder	61.44±0.00	0.52±0.00	17.97±0.00	92.07±0.00	86.93±0.00
Standard	71.59±0.05	31.81±0.29	48.41±0.27	85.56±0.20	70.09±0.19
INLP	68.54±1.05	25.13±2.31	40.70±5.02	**66.64**±0.87	66.19±0.79
Adv Single Discriminator	74.25±0.39	13.01±3.83	28.55±3.60	84.33±0.98	61.48±2.17
Adv Ensemble	74.08±0.99	12.04±3.50	31.76±3.19	85.31±0.51	63.23±3.62
Differentiated Adv Ensemble	**74.52**±0.28	**8.42**±1.84	**24.74**±2.07	84.52±0.50	**61.09**±2.32

Table 1: Evaluation results ± standard deviation (%) on the test set, averaged over 10 runs with different random seeds. **Bold** = best performance. "↑" and "↓" indicate that higher and lower performance, resp., is better for the given metric. Leakage measures the accuracy of predicting the protected attribute, over the final hidden representation $\mathbf{h}$ or model output $\hat{y}$. Since the Fixed Encoder is not designed for binary sentiment classification, we merge the original 64 labels into two categories based on the results of hierarchical clustering.

null-space, $h_M^* = P_{N(A_{\text{linear}})} h_M$, where $P_{N(A_{\text{linear}})}$ is the null-space projection matrix of A_{linear}. In doing so, it becomes difficult for the protected attribute to be linearly identified from the projected hidden representations (h_M^*), and any linear main-task classifier (C_M^*) trained on h_M^* can thus be expected to make fairer predictions.

3 Experiments

Fixed Encoder Following Elazar and Goldberg (2018) and Ravfogel et al. (2020), we use the DeepMoji model (Felbo et al., 2017) as a fixed-parameter encoder (i.e. it is not updated during training). The DeepMoji model is trained over 1246 million tweets containing one of 64 common emojis. We merge the 64 emoji labels output by DeepMoji into two super-classes based on hierarchical clustering: 'happy' and 'sad'.

Models The encoder E_M consists of a fixed pre-trained encoder (DeepMoji) and two trainable fully connected layers ("Standard" in Table 1). Every linear classifier (C) is implemented as a dense layer.

For protected attribute prediction, a discriminator (A) is a 3-layer MLP where the first 2 layers are collectively denoted as E_A, and the output layer is denoted as C_A.

TPR-GAP and TNR-GAP In classification problems, a common way of measuring bias is TPR-GAP and TNR-GAP, which evaluate the gap in the True Positive Rate (TPR) and True Negative Rate (TNR), respectively, across different protected attributes (De-Arteaga et al., 2019). This measurement is related to the criterion that the prediction $\hat{y}$ is conditionally independent of the protected attribute g given the main task label y (i.e., $\hat{y} \perp g | y$). Assuming a binary protected attribute, this conditional independence requires $\mathbb{P}\{\hat{y}|y, g = 0\} = \mathbb{P}\{\hat{y}|y, g = 1\}$, which implies an objective that minimizes the difference (GAP) between the two sides of the equation.

Linear Leakage We also measure the leakage of protected attributes. A model is said to leak information if the protected attribute can be predicted at a higher accuracy than chance, in our case, from the hidden representations the fixed encoder generates. We empirically quantify leakage with a linear support vector classifier at two different levels:

- Leakage@$\mathbf{h}$: the accuracy of recovering the protected attribute from the output of the final hidden layer after the activation function (h_M).
- Leakage@$\hat{y}$: the accuracy of recovering the protected attribute from the output $\hat{y}$ (i.e., the logits) of the main model.

Data We experiment with the dataset of Blodgett et al. (2016), which contains tweets that are either African American English (AAE)-like or Standard American English (SAE)-like (following Elazar and Goldberg (2018) and Ravfogel et al. (2020)). Each tweet is annotated with a binary "race" label (on the basis of AAE or SAE) and a binary sentiment score, which is determined by the (redacted) emoji within it.

In total, the dataset contains 200k instances, perfectly balanced across the four race–sentiment combinations. To create bias in the dataset, we follow previous work in skewing the training data to generate race–sentiment combinations (AAE–

happy, SAE–happy, AAE–sad, and SAE–sad) of 40%, 10%, 10%, and 40%, respectively. Note that we keep the test data unbiased.

Training Details All models are trained and evaluated on the same training/test split. The Adam optimizer (Kingma and Ba, 2015) is used with learning rates of 3×10^{-5} for the main model and 3×10^{-6} for the sub-discriminators. The mini-batch size is set to 1024. Sentence representations (2304d) are extracted from the DeepMoji encoder. The hidden size of each dense layer is 300 in the main model, and 256 in the sub-discriminators. We train M for 60 epochs and each A for 100 epochs, keeping the checkpoint model that performs best on the dev set. Similar to Elazar and Goldberg (2018), hyperparameters (λ_{adv} and λ_{diff}) are tuned separately rather than jointly. λ_{adv} is tuned to 0.8 based on the standard (single-discriminator) adversarial learning method, and this setting is used for all other adversarial methods. When tuning λ_{adv}, we considered both overall performance and bias gap (both over the dev data). Since adversarial training can increase overall performance while decreasing the bias gap (see Figure 2), we select the adversarial model that achieves the best task performance. For adversarial ensemble and differentiated models, we tune the hyperparameters (number of sub attackers and λ_{diff}) to achieve a similar bias level while getting the best overall performance. To compare with a baseline ensemble method with a similar number of parameters, we also report results for an adversarial ensemble model with 3 sub-discriminators. The scalar hyperparameter of the difference loss (λ_{diff}) is tuned through grid search from 10^{-4} to 10^{4}, and set to $10^{3.7}$. For the INLP experiments, fixed sentence representations are extracted from the same data split. Following Ravfogel et al. (2020), in the INLP experiments, both the discriminator and the classifier are implemented in scikit-learn as linear SVM classifiers (Pedregosa et al., 2011). We report Leakage@$\hat{y}$ for INLP based on the predicted confidence scores, which could be interpreted as logits, of the linear SVM classifiers.

Results and Analysis Table 1 shows the results over the test set. Training on a biased dataset without any fairness restrictions leads to a biased model, as seen in the Gap and Leakage results for the Standard model. Consistent with the findings of Ravfogel et al. (2020), INLP can only reduce bias at the expense of overall performance. On the other hand, the Single Discriminator and Adv(ersarial) Ensemble baselines both enhance accuracy and reduce bias, consistent with the findings of Li et al. (2018a).

Compared to the Adv Ensemble baseline, incorporating the difference loss in our method has two main benefits: training is more stable (results have smaller standard deviation), and there is less bias (the TPR and TNR Gap are smaller). Without the orthogonality factor, $\mathcal{L}_{\text{diff}}$, the sub-discriminators tend to learn similar representations, and the ensemble degenerates to a standard adversarial model. Simply relying on random initialization to ensure sub-discriminator diversity, as is done in the Adv Ensemble method, is insufficient. The orthogonality regularization in our method leads to more stable and overall better results in terms of both accuracy and TPR/TNR Gap.

As shown in Table 1, even the Fixed Encoder model leaks protected information, as a result of implicit biases during pre-training. INLP achieves significant improvement in terms of reducing linear hidden representation leakage. The reason is that Leakage@$\mathbf{h}$ is directly correlated with the objective of INLP, in minimizing the linear predictability of the protected attribute from the $\mathbf{h}$. Adversarial methods do little to mitigate Leakage@$\mathbf{h}$, but substantially decrease Leakage@$\hat{y}$ in the model output. However, both types of leakage are well above the ideal value of 50%, and therefore none of these methods can be considered as providing meaningful privacy, in part because of the fixed encoder. This finding implies that when applying adversarial learning, the pretrained model needs to be fine-tuned with the adversarial loss to have any chance of generating a truly unbiased hidden representation. Despite this, adversarial training does reduce the TPR and TNR Gap, and improves overall accuracy, which illustrates the utility of the method for both bias mitigation and as a form of regularisation.

Overall, our proposed method empirically outperforms the baseline models in terms of debiasing, with a better performance–fairness trade-off.

Robustness to λ_{adv} We first evaluate the influence of the trade-off hyperparameter λ_{adv} in adversarial learning. As can be seen from Figure 2, λ_{adv} controls the performance–fairness trade-off. Increasing λ_{adv} from 10^{-2} to around 10^{-0}, TPR Gap and TNR Gap consistently decrease, while the accuracy of each group rises. To balance up accu-

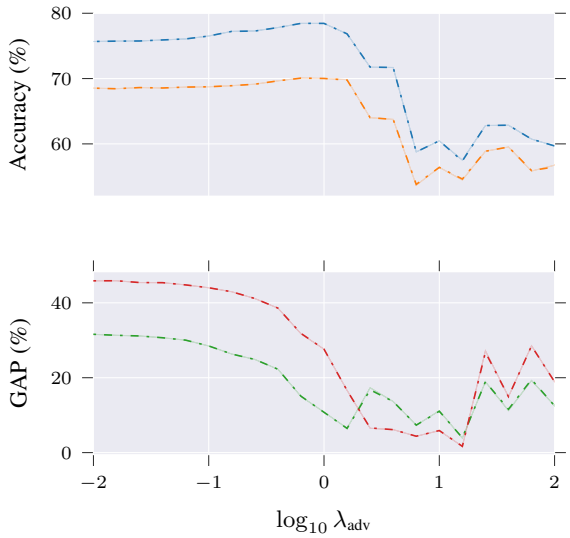

Figure 2: λ_{adv} sensitivity analysis, averaged over 10 runs for a single discriminator adversarial model. Main task accuracy of group *SAE* (blue) and *AAE* (orange), *TPR-GAP* (green), and *TNR-GAP* (red) are reported.

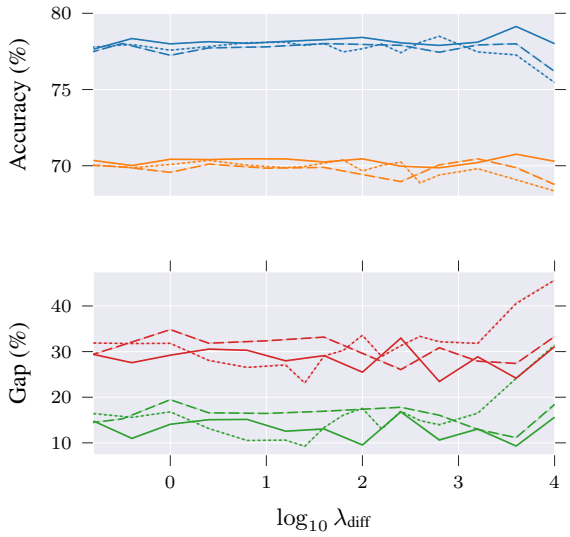

Figure 3: λ_{diff} sensitivity analysis for differentiated adversarial models with 3 ("——"), 5 ("- - - -"), and 8 ("·····") sub-discriminators, in terms of the main task accuracy of group *SAE* (blue) and *AAE* (orange), and *TPR-GAP* (green) and *TNR-GAP* (red).

racy and fairness, we set λ_{adv} to $10^{-0.1}$. We also observe that an overly large λ_{adv} can lead to a more biased model (starting from about $10^{1.2}$).

Robustness to λ_{diff} Figure 3 presents the results of our model with different λ_{diff} values, for $N \in \{3, 5, 8\}$ sub-discriminators.

First, note that when λ_{diff} is small (i.e., the left side of Figure 3), our Differentiated Adv Ensemble model generalizes to the standard Adv Ensemble model. For differing numbers of sub-discriminators, performance is similar, i.e., increasing the number of sub-discriminators beyond 3 does not improve results substantially, but does come with a computational cost. This implies that an Adv Ensemble model learns approximately the same thing as larger ensembles (but more efficiently), where the sub-discriminators can only be explicitly differentiated by their weight initializations (with different random seeds), noting that all sub-discriminators are otherwise identical in architecture, input, and optimizer.

Increasing the weight of the difference loss through λ_{diff} has a positive influence on results, but an overly large value makes the sub-discriminators underfit, and both reduces accuracy and increases TPR/TNR Gap. We observe a negative correlation between N and λ_{diff}, the main reason being that $\mathcal{L}_{\text{diff}}$ is not averaged over N and as a result, a large N and λ_{diff} force the sub-discriminators to pay too much attention to orthogonality, impeding their ability to bleach out the protected attributes.

Overall, we empirically show that λ_{diff} only needs to be tuned for Adv Ensemble, since the results for different Differentiated Adv models for a given setting achieve similar results. I.e., λ_{diff} can safely be tuned separately with all other hyperparameters fixed.

4 Conclusion and Future Work

We have proposed an approach to enhance sub-discriminators in adversarial ensembles by introducing a difference loss. Over a tweet sentiment classification task, we showed that our method substantially improves over standard adversarial methods, including ensemble-based methods.

In future work, we intend to perform experimentation over other tasks. Theoretically, our approach is general-purpose, and can be used not only for adversarial debiasing but also any other application where adversarial training is used, such as domain adaptation (Li et al., 2018b).

Acknowledgments

We thank Lea Frermann, Shivashankar Subramanian, and the anonymous reviewers for their helpful feedback and suggestions.

References

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.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain separation networks. In *Advances in Neural Information Processing Systems*, pages 343–351.

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*, FAT* '19, page 120–128.

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.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1615–1625.

Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*, pages 1180–1189.

Dirk Hovy and Anders Søgaard. 2015. Tagging performance correlates with author age. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 483–488.

Diederick P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Svetlana Kiritchenko and Saif Mohammad. 2018. Examining gender and race bias in two hundred sentiment analysis systems. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 43–53.

Yitong Li, Timothy Baldwin, and Trevor Cohn. 2018a. 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.

Yitong Li, Timothy Baldwin, and Trevor Cohn. 2018b. What's in a domain? learning domain-robust text representations using adversarial training. 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 474–479.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

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.

Mathieu Salzmann, Carl Henrik Ek, Raquel Urtasun, and Trevor Darrell. 2010. Factorized orthogonal latent spaces. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 701–708.

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.