Appendix

Debiasing Convolutional Neural Networks via Meta Orthogonalization

A Additional BAM Figures

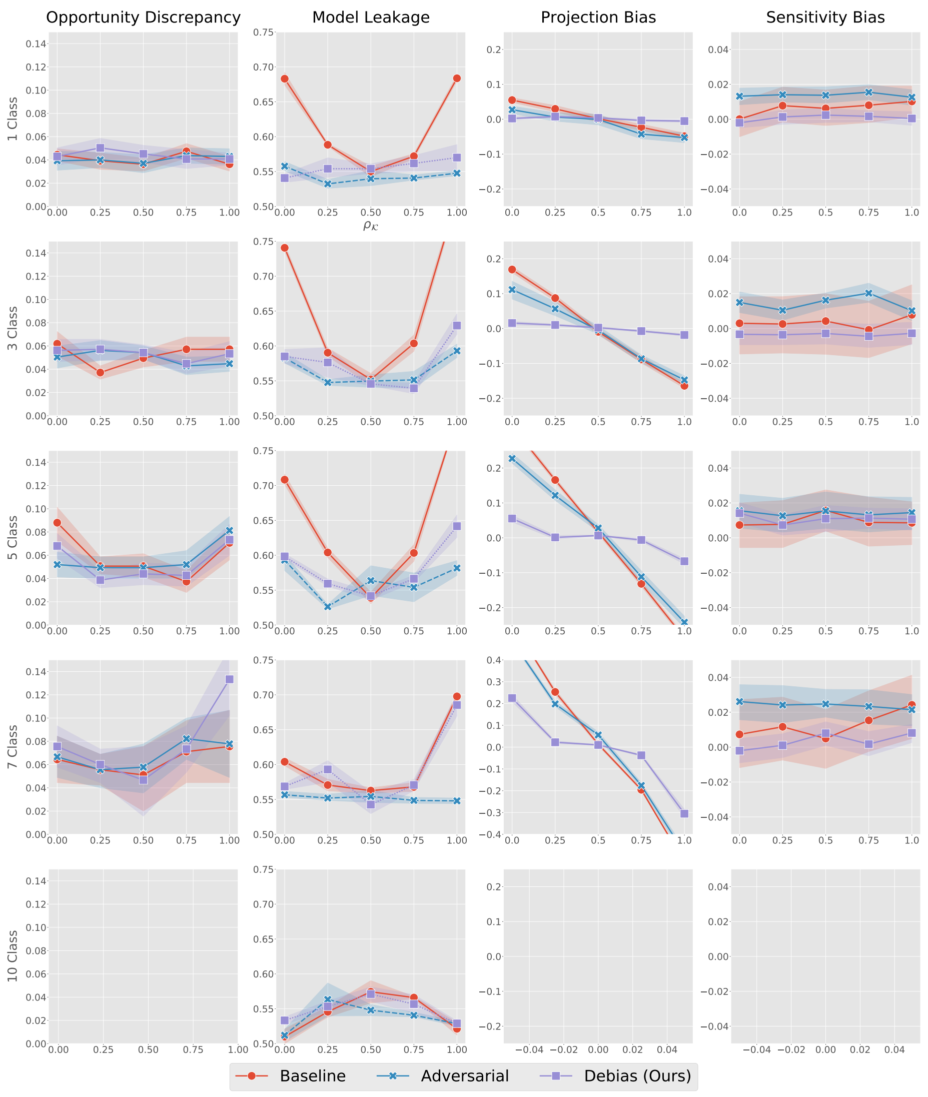

Figure A.1: Analogous to Figure 4 in the main paper; we measure the same metrics in the classes **not** contained in the biased set $\mathcal{K}$. Each have a balanced ratio of trucks/zebras.

In Figure A.1, we show the metrics when observing the classes not in the biased set. Most measures remain small and unvarying across models; only in 7 Class and Opportunity Discrepancy does ours significantly increase, for reasons described in the main paper. Model Leakage seems to maximize when 5 Classes are in $\mathcal{K}$, maybe related to some entropy measure of label co-occurences. We note an inverse correlation in projection bias that was previously observed in $\mathcal{K}$. In fact, the strength becomes larger as the biased set increases, because the co-occurence with truck/zebra becomes stronger when these classes remain balanced (since the other classes in $\mathcal{K}$ have less examples of the biased attribute. The bottom row does not contain data for the class specific metrics, because in the 10 Class case, every class is found in the biased set. Sensitivity bias remains minimal, but note that adversarially training models can result in marginally higher sensitivity.

Figure A.2 just continues Figure 4 from main paper, showing the 10 class case. It is near trivial, for similar reasons as the early ones above.

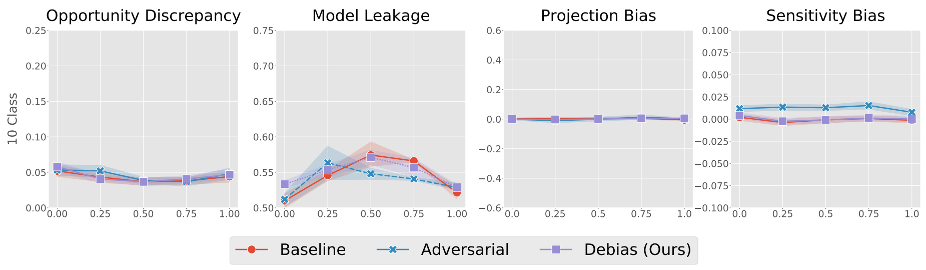

Figure A.2: Metrics across the models, when every class has the same bias ratio. We see that bias is minimal in every case, suggesting that balanced ratios within each class may not be the only way a model can be less biased (i.e. inter-class balancing could also be helpful).

B Balanced COCO Results

Table B.1: **COCO Results** ($\alpha = 1$)

Model	Discrepancy	Leakage	Projection	Sensitivity
Baseline	0.145 ±0.022	63.22*	**0.014** ±0.013	0.013 ±0.010
adv @ conv5	—	**54.91***	—	—
Debias (Ours)	**0.133** ±0.022	70.31	−0.019 ±0.014	**0.010** ±0.010

We report results on our method on the gender balanced version of COCO defined in [25]. We show decreased discrepancy, but show higher leakage and marginally different projection and sensitivity bias (since the dataset has already been balanced). * denotes obtained from [25], Table 4.

C Training Details

BAM

We now provide brief details on the whole debiasing process. We first start from an ImageNet pretrained ResNet-50, and finetune on the current dataset for 30 epochs, using SGD with a learning rate of 0.01 and momentum 0.9. We use a batch size of 64 for all runs. We then turn on $\mathcal{L}_{\text{debias}}$ while also training the class concept vectors using $\mathcal{L}_{\text{concept}}$ for 20 more epochs. The latter is trained using the same batches as the network, but with a separate SGD optimizer (without momentum, to compute the meta-step) with learning rate 0.05. To help with balancing, we subsample from the batch to reach a 50-50 split for each concept vector update. We use $\gamma = 1000$ throughout this process.

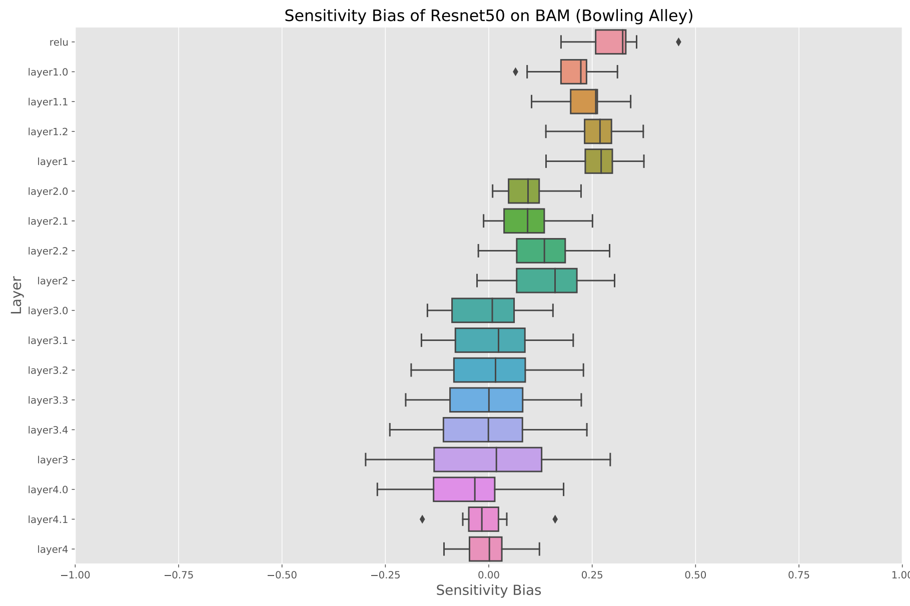

Figure C.1: Sensitivity bias at every layer in ResNet-50 for BAM, specific to the bowling alley class. We see the largest range in layer3.

A large choice is the layer to apply Meta Orthogonalization; to choose this, we take the models trained after finetuning for 30 epochs, and measure the sensitivity bias across each layer and ratio combination (assuming a single class is in the biased set). We then plot the range of these values in C.1; we use layer3 because it had the highest range of possible gradient bias for this class. For evaluation, we *retrain* the concept vectors for 25 epochs (we do not want to use the ones learned in training for evaluation) and compute the projection and sensitivity bias from these vectors. For model leakage, we train the same MLP from Wang (2018) using an Adam optimizer of learning rate 1e-5.

COCO

We follow a similar process for the COCO dataset, but instead apply $\gamma = 1\text{e}5$, model finetuning learning rate of 0.001, and a weighted binary log loss for the

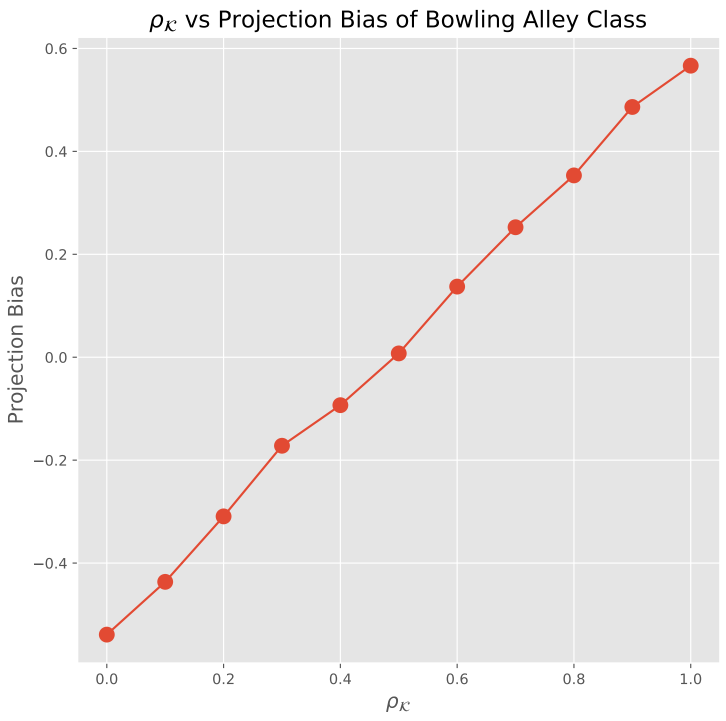

Figure C.2: Effect of $\rho_{\mathcal{K}}$ on bias projection of the bowling alley concept.

final prediction layer (based on class probabilities within the dataset). We use a batch size of 32. We also apply debias training for 30 epochs, rather than 20. For the balanced variant of COCO (i.e. Appendix B Table 1), we employ an increasing γ schedule, starting at 5e3 for 2 epochs, 1e4 for 2 epochs, and finally ending at 3e4 for the rest of training. We apply the same retraining scheme for the concept vectors but train these for 30 epochs instead. Leakage is the same, except because we saw suboptimal classification results with a learning rate of 5e-5, we tune our learning rates to achieve the highest leakage we can on our models.

D Downstream Task Performance

We lastly include the network's downstream task performance, to show that our method retains usefulness of the network. In BAM, we have better accuracy in general, except for the edge cases in 5 Class and 7 Class, which may explain the similar drop in fairness metrics. In COCO, we show marginal differences to [25].

		Ratio					
Dataset	**Model**	**0.0**	**0.25**	**0.5**	**0.75**	**1.0**	Δ
1 Class	Baseline	0.9147	0.9173	0.9263	0.9177	0.9173	**0.0039**
	Adversarial	0.8950	0.9097	0.9177	0.9083	0.8993	
	Debias	0.8980	0.9140	0.9237	0.9123	0.9017	
3 Class	Baseline	0.8943	0.9190	0.9223	0.9163	0.8960	**0.0111**
	Adversarial	0.8637	0.9000	0.9183	0.8857	0.8803	
	Debias	0.8850	0.9083	0.9190	0.9087	0.8823	
5 Class	Baseline	0.8880	0.9150	0.9173	0.9160	0.8580	−0.0098
	Adversarial	0.8640	0.8960	0.9103	0.8913	0.8740	
	Debias	0.8183	0.9003	0.9133	0.9073	0.8473	
7 Class	Baseline	0.8823	0.9160	0.9243	0.9160	0.8563	−0.0379
	Adversarial	0.8460	0.9070	0.9207	0.8970	0.8530	
	Debias	0.7787	0.9033	0.9200	0.9100	0.7220	
10 Class	Baseline	0.9053	0.9157	0.9217	0.9257	0.9040	**0.0063**
	Adversarial	0.9040	0.9000	0.9153	0.8970	0.9073	
	Debias	0.9037	0.9097	0.9223	0.9183	0.9013	

Table D.1: **BAM Accuracies** We show the classification accuracy of each model, across the situations described in the paper. The final column provides the average difference (Debias Accuracy − Adversarial Accuracy) in each dataset type.

Table D.2: **COCO Detection Results**

Model	mAP	F1
Baseline	54.05	46.92
adv@image†	53.25	47.26
Debias(Ours)	53.53	46.47

We show the corresponding mAP and F1 scores of the models we obtained for COCO. We show that ours remains competitive in this aspect as well. † denotes the model from [25]. This differs from the original as we reran with our script that may have dealt with missing object classes (in the validation set) differently.