Looking at Model Debiasing through the Lens of Anomaly Detection Supplementary Material

Vito Paolo Pastore[1]*, Massimiliano Ciranni[1]*, Davide Marinelli[1], Francesca Odone[1], Vittorio Murino[1,2,3]
[1]MaLGa, DIBRIS, University of Genoa, Italy [2]Istituto Italiano di Tecnologia, Genoa, Italy [3]University of Verona, Italy

S1. Additional Ablation Studies

S1.1 Contribution of the per-class custom threshold τ_y

In our bias identification step, we customize the OCSVM algorithm, manipulating its decision function to accommodate a custom per-class decision threshold τ_y. These thresholds (different for each class) push the anomaly detectors to reject samples as out-of-class (*i.e.*, bias-conflicting) only when they are highly confident, and thus better discriminating between bias-aligned and bias-conflicting samples. In Table S.1, we report an ablation study on Corrupted CIFAR-10 (for three considered bias correlation values ρ), evaluating the impact of using our custom thresholds or exploiting the original formulation with the default threshold (which is 0).

Corrupted CIFAR-10	$\rho = 0.995$	$\rho = 0.980$	$\rho = 0.950$
Default threshold	23.21 ± 0.07	30.80 ± 0.17	37.88 ± 0.35
Per-class custom threshold τ_y(Ours)	$\mathbf{27.26 \pm 0.47}$	$\mathbf{41.27 \pm 0.26}$	$\mathbf{50.48 \pm 0.42}$

Table S.1. Average accuracies and standard deviations (over three runs) on Corrupted CIFAR-10, where we ablate the usage of custom decision thresholds in the OCSVM anomaly detectors employed in the proposed bias identification step.

Although the default threshold can still provide reasonable debiasing performance, the proposed per-class custom thresholds lead to a significant improvement, ranging from about $+4\%$ to 13%, at the different bias levels. These results support our intuition on the necessity of obtaining a *purer* bias-conflicting prediction for improving debiasing performance.

*These authors contributed equally.
Correspondence: vito.paolo.pastore@unige.it

non-biased CIFAR-10	Accuracy
ERM+CE	84.98 ± 0.35
MoDAD	82.87 ± 0.47

Table S.2. Impact of applying MoDAD on a model trained with the original version of CIFAR-10. Reported means and standard deviations are computed from three independent runs.

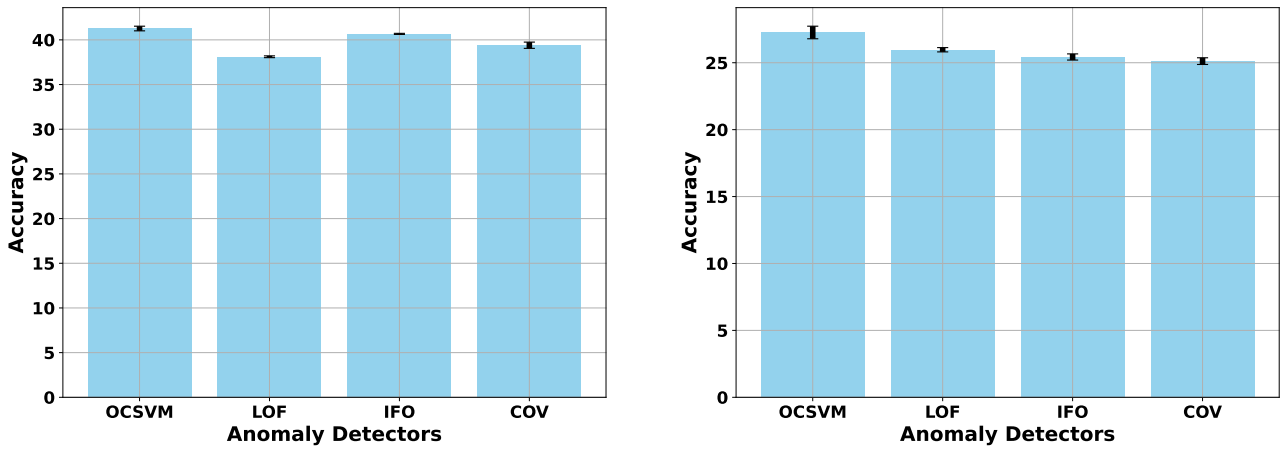
S1.2 Sensitivity to different anomaly detection algorithms

In Figure S.1, we report the impact on Corrupted CIFAR-10 in terms of average accuracy of using different anomaly detection (AD) algorithms for bias correlation values $\rho = 0.980$ and $\rho = 0.995$ (in addition to the $\rho = 0.950$ case already provided in Section 4.4 of the main paper). The comparison involves the same algorithms indicated in the main paper, i.e. OCSVM, *Local Outlier Factor* [1], *Isolation Forest* [6], and *Robust Covariance Estimator* [12].

As already noticed for the $\rho = 0.950$ case, even with a higher bias correlation, there are no significant differences in performance among the several AD methods. We notice only a slight decrease in the performance for the explored detectors, which remains in the range of 2-3% with respect to the OCSVM. This empirically proves that even when the bias correlation level increases, different anomaly detectors behave similarly.

S1.3 MoDAD impact in case of non-biased datasets

Our method (MoDAD) is designed to be applied to a biased model, *i.e.*, on a model that fails to generalize even when properly trained and tuned, suggesting the presence of bias in data or dramatic distribution shifts. In such a case, MoDAD has a significant impact, succeeding in reducing the bias effect and debiasing the model. However, in general, we do not know whether a dataset is biased or not. Hence, we are interested in assessing the effect of MoDAD when applied to an unbiased model, *i.e.*, a model trained on an unbiased dataset. Thus, we run the two steps



(a) $\rho = 0.980$.

(b) $\rho = 0.995$.

Figure S.1. MoDAD average accuracy over three different runs with different anomaly detection algorithms on Corrupted CIFAR-10.

Input Model	Average Accuracy
GCE Model	38.35 ± 0.35
Biased ERM	$\mathbf{50.48 \pm \pm 0.62}$

Table S.3. MoDAD's performance on Corrupted CIFAR10 ($\rho = 0.950$) with different input models for the debiasing step.

of MoDAD on the original non-biased version of CIFAR-10 and estimate average accuracy on the test set. As we can see in Table S.2, our debiasing method affects only slightly the performance on the test set with respect to the baseline ERM+CE trained model: we experience just a drop of $\sim 2\%$, which is significantly lower than the average benefit that we can get when using MoDAD on actually biased datasets.

S1.4 Input Model for the Debiasing Step

As reported in Section 3.3 of the main paper, our debiasing approach consists in fine-tuning a model trained with a vanilla ERM on the biased datasets. However, an alternative can be fine-tuning the intentionally biased model (trained with the GCE) utilized in the first step of our proposed approach. In Table S.3 we report a comparison between the two different input models for the debiasing step, on the Corrupted CIFAR-10 dataset (with $\rho = 0.950$). Fine-tuning the GCE model corresponds to a drop of $\sim 12\%$. This is expected, as the GCE model is intentionally and extremely biased by design, so it is more challenging to mitigate its bias dependency with respect to the biased Vanilla ERM model.

S2. Implementation Details

The code supporting this research and its experiments is implemented in `Python` [14], with the support of `Pytorch` and `Torchvision` [8, 10] for data pre-processing, neural network implementation, training, and evaluation. For the OCSVM algorithm (see Section 3 of the main paper), we rely on the open-source implementation available from `Scikit-Learn` [11]. Additionally, generic numerical operations and data visualization are performed exploiting `Numpy` [2] and `Matplotlib` [3] respectively.

S2.1 Data Pre-Processing

Input images are square-resized to a pixel resolution of 224x224 for BAR, Waterbirds, and BFFHQ. Corrupted CIFAR-10 images are instead kept at their original resolution of 32x32 pixels, following [4, 5, 9]. Regardless of the dataset, images are normalized to have RGB values between 0 and 1. Additionally, we replicate the same augmentations on the training images of Corrupted CIFAR-10, BAR, and BFFHQ that are found in [4, 5, 9], i.e.:

- **Corrupted CIFAR-10**
  1. `RandomCropping((32, 32))`
  2. `Padding((4, 4))`
  3. `RandomHorizontalFlip(p=0.5)`
- **BAR**
  1. `RandomResizedCrop((224, 224))`
  2. `RandomHorizontalFlip(p=0.5)`
  3. `ImageNet Standardization`

- **BFFHQ**
    1. `Resize((224, 224))`
    2. `Padding((4, 4))`
    3. `ImageNet Standardization`

During testing, we apply only square resizing and ImageNet standardization for BAR and BFFHQ. Concerning Waterbirds, we only perform square resizing of all input images with a target resolution of 224x224 pixels, followed by ImageNet standardization. We rely on `Torchvision` [8] for all the mentioned pre-processing operations.

S2.2 Training Details

In every experiment, we adopt AdamW as optimizer [7], with an initial learning rate of 10^{-5} for the debiasing step. For the bias-identification step involving GCE pre-training, we use an initial learning rate of 10^{-3} for Corrupted CIFAR-10 and 10^{-5} for the other datasets. The ERM models trained with Cross-Entropy loss (ERM+CE), are trained with a fixed learning rate of 10^{-3} and a mini-batch random sampler weighted on class populations. Regarding batch sizes, for the sake of fair comparisons, we follow what is found in the existing literature [5, 9, 13]: 256 for Corrupted CIFAR-10 and BAR, 128 for Waterbirds, and 64 for BFFHQ.

Network Embeddings. To extract the network embeddings employed in the proposed bias identification step (see Sec. 3.2 of the main paper), we consider the very last layer before the *softmax* layer that performs the final classification. This corresponds to the additional layer we attach to the ResNet backbone (see Training Details in Sec. 4.3 of the main paper), which is a *linear* layer with 128 neurons put after the ResNet's last *pooling* layer, followed by a ReLU non-linearity. This is fixed regardless of the dataset and the backbone being a ResNet-18 (Corrupted CIFAR-10, BAR, BFFHQ) or a ResNet-50 (Waterbirds).

Training Iterations and Regularization. The bias-identification model trained with GCE loss is trained for 100 epochs in the case of Corrupted CIFAR-10, 30 epochs for BAR and BFFHQ, and 50 epochs for Waterbirds. The Debiasing step is performed for 100 epochs in the experiments for Corrupted CIFAR-10, while 50 epochs are employed for BAR, Waterbirds, and BFFHQ. We set a fixed number of epochs and do not employ any implicit regularization technique during training, as we cannot assume datasets-wise accordance regarding distribution shifts in the validation set.

References

[1] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. Lof: Identifying density-based local outliers. *SIGMOD Rec.*, 29(2):93–104, may 2000. 1

[2] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020. 2

[3] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. 2

[4] Eungyeup Kim, Jihyeon Lee, and Jaegul Choo. Biaswap: Removing dataset bias with bias-tailored swapping augmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14992–15001, 2021. 2

[5] Jungsoo Lee, Eungyeup Kim, Juyoung Lee, Jihyeon Lee, and Jaegul Choo. Learning debiased representation via disentangled feature augmentation. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 25123–25133. Curran Associates, Inc., 2021. 2, 3

[6] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, page 413–422, USA, 2008. IEEE Computer Society. 1

[7] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. 3

[8] TorchVision maintainers and contributors. Torchvision: Pytorch's computer vision library. https://github.com/pytorch/vision, 2016. 2, 3

[9] Junhyun Nam, Hyuntak Cha, Sungsoo Ahn, Jaeho Lee, and Jinwoo Shin. Learning from failure: De-biasing classifier from biased classifier. *Advances in Neural Information Processing Systems*, 33:20673–20684, 2020. 2, 3

[10] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. 2

[11] 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. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 2

[12] Peter Rousseeuw and Katrien Driessen. A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41:212–223, 08 1999. 1

[13] Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks. In *International Conference on Learning Representations*, 2019. 3

[14] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009. 2