Learning De-biased Representations with Biased Representations – Appendix –

A. Statistical Independence is Equivalent to Functional Orthogonality for Linear Maps

We provide a proof for the following lemma in §3.2.

Lemma 1. Assume that f and g are affine mappings $f(x) = Ax + a$ and $g(x) = Bx + b$ where $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{l \times n}$. Assume further that X is a normal distribution with mean μ and covariance matrix Σ. Then, $f(X) \perp\!\!\!\perp g(X)$ if and only if $\ker(A)^{\perp} \perp_{\Sigma} \ker(B)^{\perp}$. $\perp\!\!\!\perp$ denotes the independence. For a positive semi-definite matrix Σ, we define $\langle r, s \rangle_{\Sigma} = \langle r, \Sigma s \rangle$. The orthogonality $V \perp_{\Sigma} W$ of two subspaces V and W is defined likewise.

Proof. Due to linearity and normality, the independence $f(X) \perp\!\!\!\perp g(X)$ is equivalent to the covariance condition $\mathrm{Cov}(f(X), g(X)) = 0$. The covariance is computed as:

$$\mathrm{Cov}(f(X), g(X)) = \mathbb{E}_X A(X - x^0)(X - x^0)^T B^T = A \Sigma B^T \tag{A1}$$

Note that

$$\begin{aligned} A\Sigma B^T = 0 &\iff \langle v, A\Sigma B^T w \rangle = 0 \ \forall \ v, w \iff \langle A^T v, B^T w \rangle_{\Sigma} = 0 \ \forall \ v, w \\ &\iff \mathrm{im}(A^T) \perp_{\Sigma} \mathrm{im}(B^T) \iff \ker(A)^{\perp} \perp_{\Sigma} \ker(B)^{\perp} \end{aligned} \tag{A2}$$

□

B. Algorithm

Algorithm 1 shows the detailed algorithm for solving the minimax problem in equation 3 of the main paper.

Algorithm 1 `ReBias` training

repeat
  for each mini-batch samples x, y **do**
    update f by solving $\arg\min_{f \in F} \mathcal{L}(f, x, y) + \lambda \, \mathrm{HSIC}_1(f(x), g(x))$.
    update g by solving $\arg\min_{g \in G} \mathcal{L}(g, x, y) - \lambda_g \, \mathrm{HSIC}_1(f(x), g(x))$.
  end for
until converge.

$\mathcal{L}(f, x, y)$ denotes the original loss for the main task, *e.g.*, the cross entropy loss. We solve the minimax problem in a mini-batch by employing the unbiased finite-sample estimator of HSIC, $\mathrm{HSIC}_1^{k,l}(U, V)$ (Song et al., 2012), to measure the independence of $f(X)$ and $g(X)$ in the mini-batch, defined as

$$\textbf{Unbiased HSIC estimator:} \quad \mathrm{HSIC}_1^{k,l}(U, V) = \frac{1}{m(m-3)} \left[\mathrm{tr}(\widetilde{U}\widetilde{V}^T) + \frac{\mathbf{1}^T \widetilde{U} \mathbf{1} \mathbf{1}^T \widetilde{V}^T \mathbf{1}}{(m-1)(m-2)} - \frac{2}{m-2} \mathbf{1}^T \widetilde{U} \widetilde{V}^T \mathbf{1} \right] \tag{A3}$$

C. Implementation Details

Training setup. We solve the minimax problem in algorithm 1 through alternating stochastic gradient descents with the ADAM optimiser (Kingma & Ba, 2015). The regularisation parameters λ and λ_g are set to 1.0 in all experiments. We used

the batch sizes (256, 128, 128) for (Biased MNIST, ImageNet, action recognition) experiments. For Biased MNIST, the learning rate is initially set to 0.001 and is decayed by factor 0.1 every 20 epochs. For ImageNet and action recognition, learning rates are initially set to 0.001 and 0.1, respectively, and are decayed by cosine annealing. For action recognition, we use $f(x)$ and $g(x)$ as the output logits for the sake of stable training. For each dataset, we train every method (vanilla, biased, comparison methods, and `ReBias`) for the same number of epochs. We train the models with (80, 120, 120) epochs for (Biased MNIST, ImageNet, action recognition) experiments. All experiments are implemented using PyTorch (Paszke et al., 2019).

In addition, we observe that a better optimiser than ADAM, *e.g.* AdamP (Heo et al., 2020), helps performance improvements in Biased MNIST and ImageNet experiments. Table A1 shows the results from the reference paper (Heo et al., 2020). For future researches, we recommend using AdamP for better optimisation stability.

Table A1. **AdamP experiments.** Biased MNIST and 9-Class ImageNet benchmarks with AdamP (Heo et al., 2020).

Optimizer	Biased MNIST Unbiased acc. at ρ					9-Class ImageNet		
	.999	.997	.995	.990	avg.	Biased	UnBiased	IN-A
Adam (Kingma & Ba, 2015)	22.9	63.0	74.9	87.0	61.9	93.8	92.6	31.2
AdamP (Heo et al., 2020)	**30.5** (+7.5)	**70.9** (+7.9)	**80.9** (+6.0)	**89.6** (+2.6)	**68.0** (+6.0)	**95.2** (+1.4)	**94.5** (+1.8)	**32.9** (+1.7)

Architecture details for action recognition. The `3D-ResNet` (Tran et al., 2019; Feichtenhofer et al., 2019) architecture is widely used in various video understanding tasks. We choose `3D-ResNet18` and `2D-ResNet18` as F and G of our method, respectively. The architectural details are in Table A2.

Training details for comparison methods. For training `LearnedMixin`, we pre-train and fix G before training F, as done in the original paper. We pre-train G for 5 epochs for Biased MNIST, and 30 epochs for ImageNet and action recognition. For training `RUBi`, we update F and G simultaneously without pre-training G, as done in the original paper. For training `HEX`, we substitute G with the neural grey-level co-occurrence matrix (NGLCM) to represent the "superficial statistics". For `StylisedImageNet`, we augment 9-class ImageNet with its stylised version (*i.e.*, twice the original dataset size), while maintaining the training setup as identical.

D. Standard Errors in Experimental Results

We report the standard errors of the main paper results in Table A3 (Biased MNIST), Table A4 (ImageNet) and Table A5 (action recognition).

E. Decision Boundary Visualisation for Toy Experiment

We show the decision boundaries of the toy experiment (§3.2) in Figure A1.

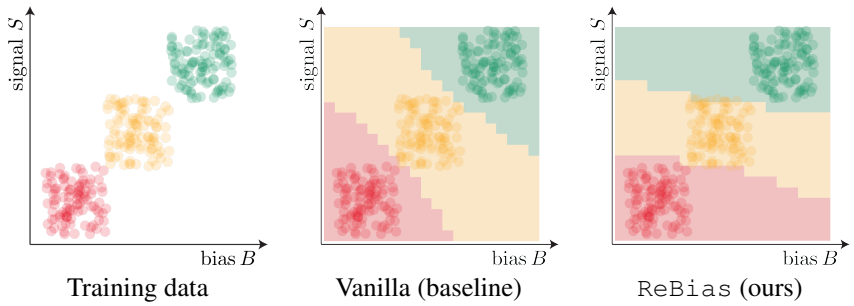


Figure A1. **Decision boundaries of toy data**. GIF animations are at anonymous link.

layer name	3D-ResNet18	2D-ResNet18	output size (T×C×H×W)
input	input video	input video	8×3×224×224
conv1	**3**×7×7, 32 stride=2	**1**×7×7, 32 stride=2	8×32×112×112
pool1	1×3×7, max stride=2	1×3×7, max stride=2	8×32×56×56
resblock2	[**3**×3×3, 32 1×3×3, 32] ×2	[**1**×3×3, 32 1×3×3, 32] ×2	8×32×56×56
resblock3	[**3**×3×3, 64 1×3×3, 64] ×2	[**1**×3×3, 32 1×3×3, 32] ×2	8×64×28×28
resblock4	[**3**×3×3, 128 1×3×3, 128] ×2	[**1**×3×3, 32 1×3×3, 32] ×2	8×128×14×14
resblock5	[**3**×3×3, 256 1×3×3, 256] ×2	[**1**×3×3, 32 1×3×3, 32] ×2	8×256×7×7
final	global avg. pool fc	global avg. pool fc	# classes

Table A2. **Network architecture for action recognisers.** The kernel dimensions of convolution layers are denoted as $[\mathrm{T} \times \mathrm{W} \times \mathrm{H}, \mathrm{C}]$ along temporal (T), width (W), height (H), and channel (C) axes, respectively. The main difference between `3D-ResNet18` and `2D-ResNet18` is the temporal kernel size.

ρ	Vanilla	Biased	HEX	LearnedMixin	RUBi	ReBias (ours)
.999	10.4 ± 0.5	10. ± 0.	10.8 ± 0.4	12.1 ± 0.8	13.7 ± 0.7	**22.7** ± 0.4
.997	33.4 ± 12.1	10. ± 0.	16.6 ± 0.8	50.2 ± 4.5	43.0 ± 1.1	**64.2** ± 0.8
.995	72.1 ± 1.9	10. ± 0.	19.7 ± 1.9	78.2 ± 0.7	**90.4** ± 0.4	76.0 ± 0.6
.990	89.1 ± 0.1	10. ± 0.	24.7 ± 1.6	88.3 ± 0.7	**93.6** ± 0.4	88.1 ± 0.6
avg.	51.2	10.	18.0	57.2	60.2	**62.7**

Table A3. **Unbiased accuracy on Biased MNIST.** Unbiased accuracies on varying train correlation ρ. Means and standard errors are computed over three independent runs.

Model description	Biased	Unbiased	IN-A
Vanilla (`ResNet18`)	90.8 ± 0.6	88.8 ± 0.6	24.9 ± 1.1
Biased (`BagNet18`)	67.7 ± 0.3	65.9 ± 0.3	18.8 ± 1.1
`StylisedIN` (Geirhos et al., 2019)	88.4 ± 0.5	86.6 ± 0.6	24.6 ± 1.4
`LearnedMixin` (Clark et al., 2019)	64.1 ± 4.0	62.7 ± 3.1	15.0 ± 1.6
`RUBi` (Cadene et al., 2019)	90.5± 0.3	88.6 ± 0.4	27.7 ± 2.1
`ReBias` (ours)	**91.9** ± 1.7	**90.5** ± 1.7	**29.6** ± 1.6

Table A4. **ImageNet results.** We show results corresponding to $F =$ `ResNet18` and $G =$ `BagNet18`. IN-A indicates ImageNet-A. Means and standard errors are computed over three independent runs.

Model description	Biased (Kinetics)	Unbiased (Mimetics)
Vanilla (`3D-ResNet18`)	54.5 ± 3.2	18.9 ± 0.4
Biased (`2D-ResNet18`)	50.7 ± 3.3	18.4 ± 2.3
`LearnedMixin` (Clark et al., 2019)	12.3 ± 2.3	11.4 ± 0.4
`RUBi` (Cadene et al., 2019)	22.4 ± 2.0	13.4 ± 1.5
`ReBias` (ours)	**55.8** ± 3.1	**22.4** ± 1.3

Table A5. **Action recognition results.** We show results corresponding to $F =$ `3D-ResNet18` and $G =$ `2D-ResNet18` with baseline comparisons. Top-1 accuracies are reported. Means and standard errors are computed over three independent runs.

F. Texture Clustering on ImageNet

In our ImageNet experiments (§4.3), we have obtained the proxy ground truths for texture bias using texture feature clustering. We extract the texture features from images by computing the gram matrices of low-layer feature maps, as done in texturisation methods (Gatys et al., 2015; Johnson et al., 2016), to capture the edge and colour cues. Specifically, we use the feature maps from layer `relu1_2` of the ImageNet pre-trained `VGG16` (Simonyan & Zisserman, 2015).

To approximate the texture ground truth labels, we cluster the texture features of 9-Class ImageNet data. We use the mini-batch k-means algorithm with $k = 9$ and batch size 1024. As k-means clustering is non-convex, we repeat the ImageNet experiments with three different texture clustering results, each with different initialisation. We report the averaged performances across the three trials.

We show an example texture clustering in Figure A2. Clusters capture similar texture patterns. The texture clusters exhibit strong correlations with semantic classes. See Figure A3 for the top-3 correlated classes per cluster. For example, the "water" texture is strongly associated with the turtle, fish, and bird classes. In the presence of such bias-class correlations, the model is motivated to take the bias shortcut by utilising the texture cue for recognition. In this case, the model shows sub-optimal performances on unusual class-texture combinations (*e.g.*, crab on grass texture).

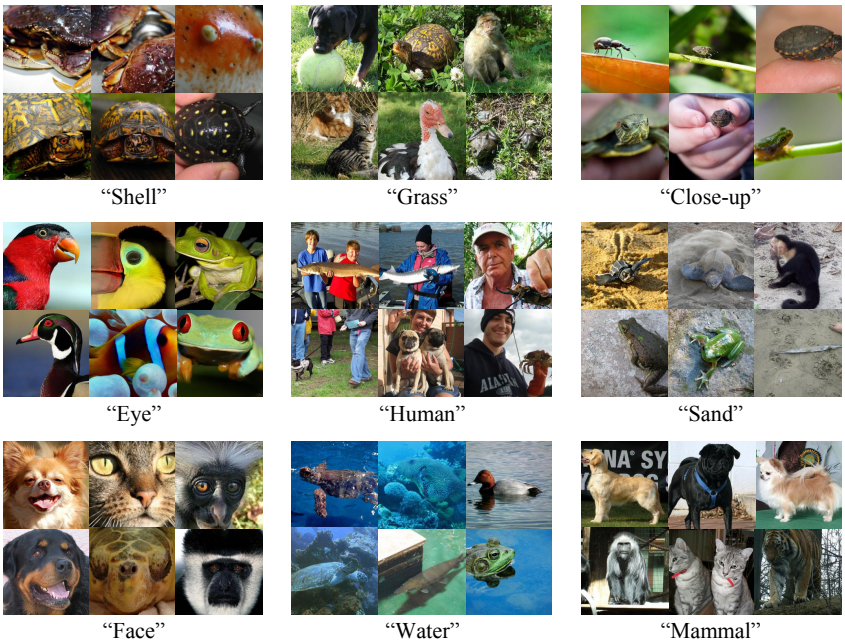
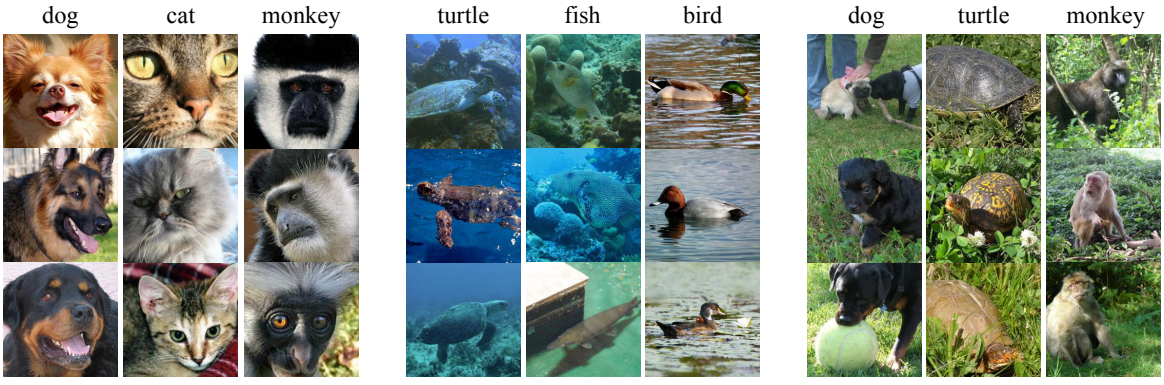


Figure A2. **Texture clusters.** Example images from texture clusters. Each cluster is named according to the common texture pattern.



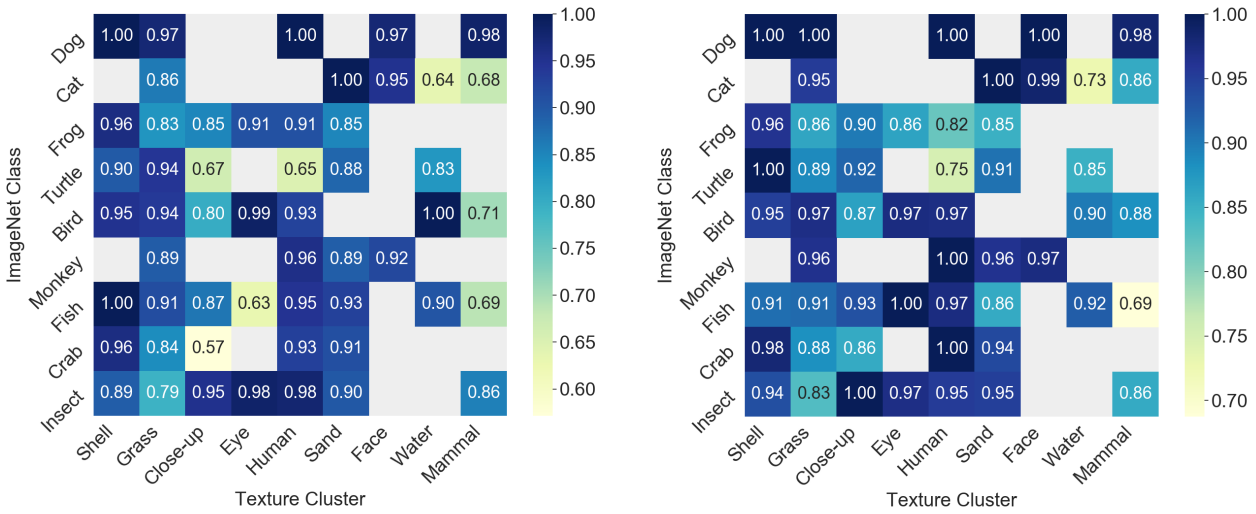
(a) "Face" cluster samples (b) "Water" cluster samples (c) "Grass" cluster samples

Figure A3. **Dominant texture-class correlations.** For each cluster, we visualise its top-3 correlated classes.

G. Further Analysis on Texture Bias of ImageNet-Trained Models

We further analyse the texture bias of the models trained on ImageNet (§4.3). Figure A4 shows the texture-class-wise accuracies of the vanilla-trained `ResNet18` and `ReBias`-trained `ResNet18`. To quantitatively measure the texture biases, we count the number of samples for each texture-class pair (c, y) ("population" Pop(c, y) in the main paper). For each class, we define the *dominant texture cluster* as the largest cluster that contains $> 30\%$ of the class samples. 4 out of 9 classes have the dominant texture cluster: ("Dog", "Face"), ("Cat", "Face"), ("Bird", "Eye"), and ("Monkey", "Face").

We measure the average accuracy over classes with *dominant texture clusters* (biased classes) and the average over the other classes. We observe that `ResNet18` shows higher accuracy on biased classes (90.6%) than on less biased classes (86.3%), signifying its bias towards texture. On the other hand, `ReBias` achieves similar accuracies 94.8% (biased classes) and 90.4% (unbiased classes). We stress that `ReBias` overcomes the bias and enhances generalisation across distributions even if the training dataset itself is biased.



(a) Vanilla trained `ResNet18`.

(b) `ReBias` trained `ResNet18`.

Figure A4. **Texture-class-wise accuracies on ImageNet.** For every texture-class pair $(B, Y) = (b, y)$, corresponding accuracy is visualised. We ignore cells with population less than 10 (masked in gray).

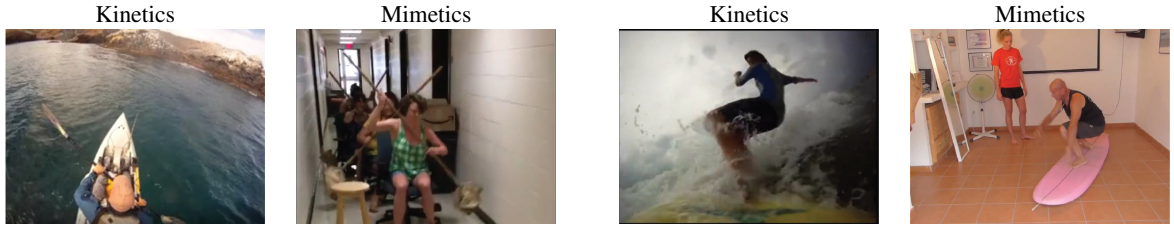


Figure A5. **Kinetics and Mimetics datasets.** We show examples of two classes, "canoeing or kayaking" and "surfing water". Kinetics samples are biased towards certain scene contexts (*e.g.* ocean), but Mimetics is relatively free from such context biases.

H. Action Recognition Datasets

We provide an overview of the action recognition datasets. Kinetics dataset (Carreira & Zisserman, 2017) has 300K videos with 400 action classes. Mimetics dataset (Weinzaepfel & Rogez, 2019) has 713 videos with 50 action classes (subset of the Kinetics classes). To simplify our investigation, we have sub-sampled 10 common classes from Kinetics and Mimetics: *"canoeing or kayaking", "climbing a rope", "driving car", "golf driving", "opening bottle", "playing piano", "playing volleyball", "shooting goal (soccer)", "surfing water", and "writing"*. Examples of Kinetics and Mimetics are shown in Figure A5. While Kinetics samples are biased towards static cues like scene and objects, Mimetics are relatively free of such correlations. Mimetics is thus a suitable benchmark for validating the cross-bias generalisation performances.

References

Cadene, R., Dancette, C., Cord, M., Parikh, D., et al. Rubi: Reducing unimodal biases for visual question answering. In *Advances in Neural Information Processing Systems*, pp. 839–850, 2019.

Carreira, J. and Zisserman, A. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6299–6308, 2017.

Clark, C., Yatskar, M., and Zettlemoyer, L. Don't take the easy way out: Ensemble based methods for avoiding known dataset biases. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4069–4082, 2019.

Feichtenhofer, C., Fan, H., Malik, J., and He, K. Slowfast networks for video recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 6202–6211, 2019.

Gatys, L., Ecker, A. S., and Bethge, M. Texture synthesis using convolutional neural networks. In *Advances in neural information processing systems*, pp. 262–270, 2015.

Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F. A., and Brendel, W. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bygh9j09KX.

Heo, B., Chun, S., Oh, S. J., Han, D., Yun, S., Uh, Y., and Ha, J.-W. Slowing down the weight norm increase in momentum-based optimizers. *arXiv preprint arXiv:2006.08217*, 2020.

Johnson, J., Alahi, A., and Fei-Fei, L. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pp. 694–711. Springer, 2016.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

Song, L., Smola, A., Gretton, A., Bedo, J., and Borgwardt, K. Feature selection via dependence maximization. *Journal of Machine Learning Research*, 13(May):1393–1434, 2012.

Tran, D., Wang, H., Torresani, L., and Feiszli, M. Video classification with channel-separated convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5552–5561, 2019.

Weinzaepfel, P. and Rogez, G. Mimetics: Towards understanding human actions out of context. *arXiv preprint arXiv:1912.07249*, 2019.