

DeepTechnome: Mitigating Unknown Bias in Deep Learning Based Assessment of CT Images

Simon Langer¹, Oliver Taubmann², Felix Denzinger^{1,2}, Andreas Maier¹, and Alexander Mühlberg²

¹ Pattern Recognition Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

{simon.langer, felix.denzinger, andreas.maier}@fau.de

² Siemens Healthcare GmbH, Forchheim, Germany

oliver.taubmann@siemens-healthineers.com, alexander-muehlberg@hotmail.com

Abstract. Reliably detecting diseases using relevant biological information is crucial for real-world applicability of deep learning techniques in medical imaging. We debias deep learning models during training against unknown bias – without preprocessing/filtering the input beforehand or assuming specific knowledge about its distribution or precise nature in the dataset. We use control regions as surrogates that carry information regarding the bias, employ the classifier model to extract features, and suppress biased intermediate features with our custom, modular *DecorreLayer*. We evaluate our method on a dataset of 952 lung computed tomography scans by introducing simulated biases w.r.t. reconstruction kernel and noise level and propose including an adversarial test set in evaluations of bias reduction techniques. In a moderately sized model architecture, applying the proposed method to learn from data exhibiting a strong bias, it near-perfectly recovers the classification performance observed when training with corresponding unbiased data.

Keywords: DL · Unknown Bias · Debiasing · CT · Technical Variation

1 Introduction

Similarly to the batch effect in genomics [15], technical variation in CT scans occurs for a variety of reasons, becoming especially problematic when it is correlated with the predictive task, for instance due to prior knowledge of the clinician and/or patient of a likely diagnosis, or site-specific differences in patient selection and acquisition protocols within multi-center data sets [16].

The range of reconstruction and scan parameters affects the amount and appearance of technical variation present [7,19]: choosing an appropriate reconstruction kernel forces a tradeoff between detail and noise level; tube current and voltage affect the amount of noise and radiation dose. The severity and presence of beam hardening, scatter, metal, motion and truncation artifacts are influenced by such choices as well as by patient behavior and physiology.

When automating image analysis with deep/machine learning (DL/ML), bias poses a fundamental challenge – consequently, there has been substantial research and industry interest in it. When tackling known bias, normalizing the input data w.r.t. technical variation in a preprocessing step is a classical approach; a recent example employs DL to preprocess images, converting their style using convolutional neural networks which predict the image differences of reconstruction kernels [6]. Alternatively, adversarial debiasing can be performed, i.e. a secondary network debiases while training using gradient reversal and/or min-max based approaches [1,12,20]. In contrast, there has been relatively little work in DL tackling the more general, universal case of unknown bias. Notably, a resampling preprocessing approach accounts for less frequent permutations of image properties – acquired with a variational autoencoder (VAE) [13] – by sampling the associated images more frequently [2]. In classical ML, RAVEL regresses unwanted features per voxel using control regions [8] and the Technome [16] combines debiasing and training in order to avoid the risk of removing informative biological information when stabilizing during preprocessing. Our work aims to transfer such ideas to the field of DL.

2 Method

Control regions (CRs) serve as surrogates, capturing the technical variation but little to no biological variation related to the detection task. To maximize similarity e.g. in streak and noise patterns, areas of the CT scan which are outside of, but close to the regions of interest (ROIs) such as surrounding air or anatomical structures are good candidates; the designer chooses depending on the presumed general types of biases that might be present. This introduces the assumption that the confounder manifests not only in the task-relevant area, yet no other knowledge about the specific nature or distribution of the bias is required.

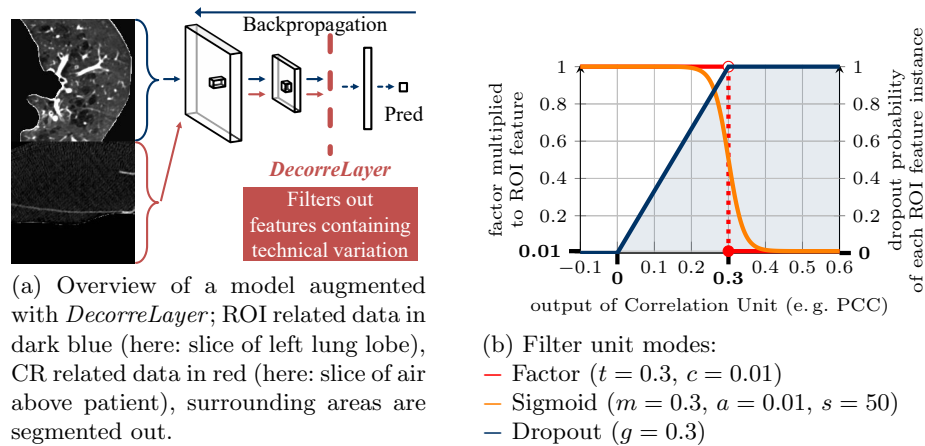


Fig. 1. *DecorreLayer* overview (a) and details on its Filter Unit module (b).

Our novel, modular *DecorreLayer* is inserted into an existing classifier model, receiving features calculated by the model architecture up to that point for both the ROI and CR (applying the same computations to ROI and CR data), and returns a filtered version of the ROI features, as illustrated in Fig. 1a.

2.1 Correlation Unit

DecorreLayer consists of two modules with separate tasks. The Correlation Unit determines what features probably depend on technical variation, comparing each feature with its control region counterpart over the batch dimension. For the sake of simplicity, the Pearson correlation coefficient (PCC) was used for this purpose in our experiments. In order to operate independently of the location of e.g. a noise pattern, a global average pooled virtual feature vector (one scalar feature as the mean over each pixel in a channel) is used if *DecorreLayer*'s output will still be interpreted with spatial information (e.g. by a subsequent convolutional layer). While the use of PCC may appear to limit the approach to linear co-dependencies, note that it is computed on internal feature maps that are themselves non-linear w.r.t. the input data. Nonetheless, it could also be replaced with other correlation measures, linear or non-linear, if deemed appropriate – even trainable ones. We suggest inserting *DecorreLayer* prior to every fully connected and convolutional layer, except the first one.

2.2 Filter Unit

The Filter Unit is in direct contact with the surrounding architecture. It generates filtered versions of the ROI features as *DecorreLayer*'s output, with stronger filtering being applied when the Correlation Unit's output is higher (Fig. 1b): Either by multiplying with a small constant c if the PCC is above a threshold t (*Factor* mode), a smoothed variant with middle m , minimum a and steepness s (*Sigmoid* mode), or by interpreting dependency on technical variation as a dropout probability, guaranteeing dropout at $\geq g$ (*Dropout* mode). The latter performed best, gradually reducing the reliability of a feature the more it is probably dependent on technical variation by increasing the dropout probability appropriately. With $\text{Bern}(p)$ being the Bernoulli distribution, \mathbf{x} denoting all instances of one feature over the batch dimension and \hat{d} the scalar PCC, the result $\hat{\mathbf{y}}$ can be described as¹:

$$\mathbf{z} \sim \text{Bern} \left(\max \left\{ 0, 1 - \max \left\{ 0, \hat{d} \right\} \cdot g^{-1} \right\} \right) \quad (1)$$

$$\hat{\mathbf{y}} = \mathbf{x} \cdot \mathbf{z}. \quad (2)$$

¹ For the sake of readability, we use a mix of regular and random variables.

2.3 Backward Pass and Inference

We intentionally relax the constraint of perfectly matching the forward pass computations during the backward pass: *DecorreLayer* is ignored entirely, which results in a reweighting of the error tensor such that errors caused by *DecorreLayer*’s filtering of features containing technical variation are “blamed on” the previous layers. At inference time, *DecorreLayer* is inactive since the model has already learned to extract features without bias, i. e. CRs are no longer needed.

3 Data Preparation and Testing Setup

Our evaluation data set contains 952 lung CT scans from a single site, acquired with a SOMATOM[®] Force and labeled for the presence (label **CE**) or absence (**nCE**) of centrilobular emphysema. As this disease is primarily visible in the upper half of the lung [3], we extract 5 evenly spaced lung-masked axial slices from that area as ROIs, and unrelated air regions above the patient from the same slices as CRs (cf. Fig. 1a). Since each scan was reconstructed with both a soft (Br36d3) and a sharp (Bl57d3) kernel, we can simulate realistic technical variation by more frequently sampling softer images for label **CE**, and otherwise sharper, but noisier images ($\text{CE} \leftrightarrow 90\%$ chance of soft image). In a separate experiment with artificial technical variation we apply additive white Gaussian noise (AWGN, $\mu = 0$, $\sigma = 0.5 \sigma_{\text{ROI}}$, σ_{ROI} denotes the standard deviation of the ROI voxel intensities) to images labeled **CE** with a 90% chance.

We define *adversarial test sets* as an inversion of the introduced manipulations, affecting every image in the test set, and argue for their adoption in future work on debiasing DL/ML since they directly display worst-bias-case performance as well as visualize the reliance of a model on technical variation (such as extra noise) not present in the original, unbiased data (*full test set*).

We also propose the Histogram of Correlations – a visualization technique which plots the correlations of ROI and CR activations (i. e. the Correlation Unit output), optionally including a comparison with unbiased and/or debiased trainings (cf. Fig. 2). It both facilitates finding a reasonable hyperparameter range for *DecorreLayer*, and investigating our or other debiasing techniques.

Our approach is tested on three architectures: A) a small custom model consisting of 2 sets of convolutional, ReLU, MaxPool layers (6, 16 channels) followed by 3 fully connected ones, B) a medium custom model consisting of 3 wider sets of this structure (32, 64, 128 channels), followed by 4 fully connected layers, as well as C) ResNet-18 [11] with batch normalization disabled; training is performed with minimal augmentations (horizontal flip, translation) using stochastic gradient descent.

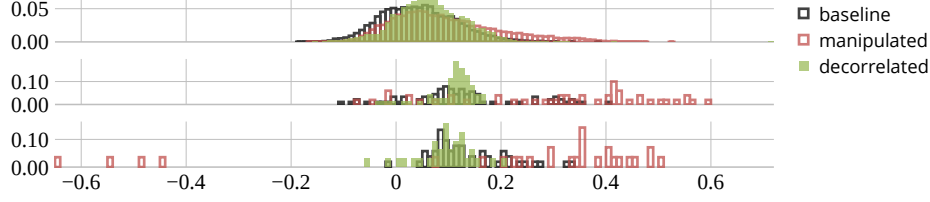


Fig. 2. Histogram of Correlations with 3 *DecorreLayers* inserted at depths 6, 9, 11 (their individual histograms shown top to bottom) into the Small Custom Architecture. We visualize how many activations exhibit a specific (here: Pearson) correlation value (bins on x-axis), normalized by total number of activations (y-axis). Note the shift towards positive correlation when training a model on manipulated, i.e. biased data, compared to the baseline – *DecorreLayer* successfully reverts this effect.

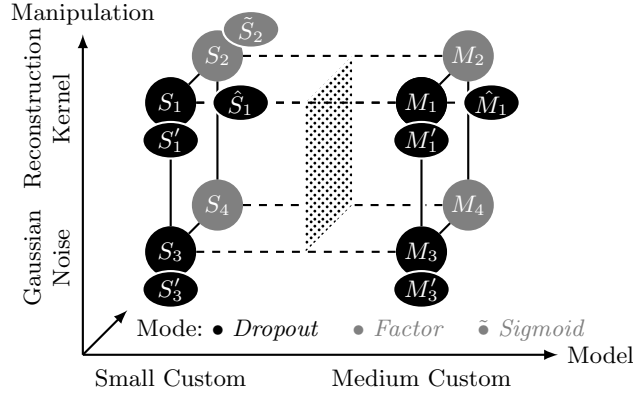


Fig. 3. Visualization of core evaluations of our method. We keep hyperparameters of nodes connected with solid lines equal (except for the different Filter Unit modes). When fully switching architecture or changing it by e.g. introducing batch normalization, we need to adapt the hyperparameters – nodes connected with dashed lines are only semantically similar. An apostrophe (') denotes evaluations with weakened bias (from 0.9 to 0.7), a hat (^) architectures with batch normalization, and a tilde in a gray node (~) trainings with the *Sigmoid* instead of the *Factor* mode. Analogous nodes for ResNet (R_3 and R_4) were omitted to improve readability.

4 Evaluation and Discussion

The mean ROC-AUC of a 5-fold cross validation serves as the main evaluation metric; Fig. 3 visualizes and explains the test setups represented by abbreviations such as S_1 . The results are summarized in Table 1 as the ROC-AUCs after half of the (typically over-)allocated training time of 300 epochs.

In Fig. 4 (S_1), we demonstrate the inflated score one would see when e.g. training and testing the small custom architecture on data with reconstruction kernel bias (ROC-AUC .930), compared to the dramatic impact on true (.587 ROC-

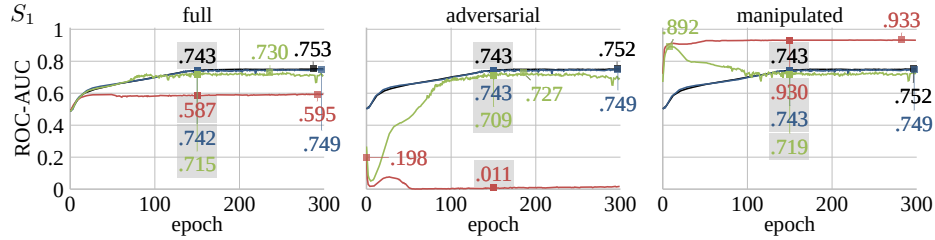


Fig. 4. *DecorreLayer* on the small custom architecture. The subplots represent the different test sets: full (unbiased), adversarial, and manipulated (same bias as training data). Color coding denotes whether training data was biased (□/■) or unbiased (□/■) and *DecorreLayer* was enabled or disabled (solid ■/■ vs. empty □/□ box).

AUC) and especially adversarial performance (.011 ROC-AUC). *DecorreLayer* minimizes this issue, achieving scores within .035 ROC-AUC of the baseline using the recommended *Dropout* mode of the Filter Unit.

DecorreLayer also effectively decorrelates when applied to weaker reconstruction kernel bias (S'_1) and/or incorporated into the medium custom architecture (M_1, M'_1). Further tests of AWGN added to most images of ill patients ($S_3, S'_3, S_4, M_3, M'_3, M_4$) show that hyperparameters e.g. gained from another type of bias can serve as a reasonable basis for new trainings. The performance of *Factor* (S_2, M_2) or *Sigmoid* (\hat{S}_2) Filter Unit modes is typically worse, and their decorrelation effect is less sustained – when examining the full plots, (especially adversarial) performance tends to drop again in favor of re-learning the bias.

Applying *DecorreLayer* to the comparatively huge ResNet-18 did not work as easily (R_3); further investigation would be necessary to determine if *DecorreLayer* can work consistently on ResNet-18 with dataset sizes common in medical applications. We decorrelated a training on AWGN manipulated data using the *Factor* mode (R_4) at the cost of degraded performance if no bias had been present. However, we observed promising results when performing additional testing with ResNet-18 on CIFAR-10 [14] (cf. supplementary material, Figs. 1 and 2). Also, *DecorreLayer* is currently incompatible with batch normalization; we argue that the mean and variance estimates of batch normalization might fluctuate significantly due to the changing number of filtered features, hence creating conflicts.

Having determined appropriate hyperparameters for *DecorreLayer* using the proposed Histogram of Correlations, it has very little negative – and sometimes even a positive – impact on training stability. For instance, we observed occasional stability issues causing divergence to 0.5 ROC-AUC when training our medium custom architecture on biased data; in this case, *DecorreLayer* acted in a regularizing fashion like a regular dropout layer [18]. Furthermore, we still consider access to a small unbiased data set vital for real-world application: to determine suitable hyperparameters and gain interpretable testing results.

Table 1. Model performance after 150 training epochs. Each column represents a model, with which we performed 4 trainings; when training on biased data, both the unbiased as well as adversarial ROC AUCs are almost always much higher when employing *DecorreLayer* (green filled square) in contrast to not performing any debiasing (red empty square). The respective standard deviations are provided in the supplementary material.

Test	Train+Decorr		Small Custom Architecture							
	Decor on ■ / ■		S_1	S'_1	S_2	\tilde{S}_2	S_3	S'_3	S_4	\hat{S}_1
unbiased	unbiased		.743	.743	.743	.743	.743	.743	.743	.740
			.742	.742	.739	.738	.742	.742	.739	.749
	biased		.587	.663	.587	.587	.611	.703	.611	.656
			.715	.725	.714	.706	.674	.701	.686	.592
	p -value		.001	.001	.001	.003	.075	.810	.051	.203
adversarial	unbiased		.743	.743	.743	.743	.743	.743	.743	.742
			.743	.743	.740	.739	.743	.743	.740	.750
	biased		.011	.286	.011	.011	.011	.314	.011	.293
			.709	.744	.644	.607	.655	.596	.622	.543
	p -value		.000	.000	.000	.000	.000	.000	.000	.107

Test	Train+Decorr		Medium Custom Architecture							ResNet-18	
	Decor on ■ / ■		M_1	M'_1	M_2	M_3	M'_3	M_4	\hat{M}_1	R_3	R_4
unbiased	unbiased		.731	.731	.731	.731	.731	.731	.730	.740	.740
			.703	.703	.752	.703	.703	.752	.710	.557	.614
	biased		.554	.634	.554	.574	.670	.574	.626	.645	.645
			.713	.677	.554	.667	.711	.726	.617	.570	.699
	p -value		.006	.343	.998	.065	.357	.016	.846	.007	.041
adversarial	unbiased		.729	.729	.729	.729	.729	.729	.729	.739	.739
			.707	.707	.747	.707	.707	.747	.710	.558	.618
	biased		.240	.343	.240	.492	.529	.492	.224	.016	.016
			.749	.725	.204	.667	.711	.719	.272	.711	.652
	p -value		.013	.020	.802	.119	.161	.033	.679	.000	.000

5 Conclusion and Outlook

To the best of our knowledge, *DecorreLayer* is the first method to debias DL models while training on data containing unknown biases: we teach models not to use bias present in training data as long as the confounder is also observable in image areas that are not directly related to the task at hand. While currently working less reliably on larger architectures, as well as requiring hyperparameter values carefully adjusted using the proposed tools, in our experiments on CT lung emphysema classification *DecorreLayer* was able to immensely boost generalization performance under adverse conditions of both artificially created and intentionally sampled technical variation – often near-perfectly recovering the baseline performance while training on heavily biased data.

To achieve this, instances of *DecorreLayer* are inserted into the model pipeline, they analyze how the model perceives control regions in addition to the regions of interest. It does not entail e. g. an additional loss term, but integrates into the architecture as a smart filtering/regularization layer. During training, computational requirements increase somewhat due to an additional forward pass for the control region. In exchange, *DecorreLayer* introduces no trainable parameters since it re-uses the feature extraction of the main model and does not create any load when performance matters most, i. e. during inference – the original architecture can perform its task without extra help. We argue that slightly relaxing the correctness of the gradient calculation can provide new ways to steer training into the intended direction.

Future work directly expanding on *DecorreLayer* could learn/dynamically calculate Filter Unit hyperparameters, or investigate new Correlation Units: A) applying other existing measures of dependence like the Hilbert-Schmidt independence criterion [9] or mutual information [17] and, B) learning an arbitrary correlation function, which could circumvent the risk of models bypassing correlation checks by considering earlier feature maps (or even unprocessed CRs) to compare ROI features against. Future work also needs to investigate how well the approach generalizes to various modalities as well as other types of bias. We expect long-term developments to head towards more directly informing the main architecture of what it did wrong when learning biased features: for instance by combining techniques limited to known bias like adversarial debiasing with un- or self-supervised techniques such as VAEs [13], or SimCLR(v2) [4,5] and BYOL [10] to generate representations of technical variation.

References

1. Alvi, M., Zisserman, A., Nellaker, C.: Turning a blind eye: Explicit removal of biases and variation from deep neural network embeddings. *ECCV Workshops* **5**, 556–572 (2018)
2. Amini, A., Soleimany, A., Schwarting, W., Bhatia, S., Rus, D.: Uncovering and mitigating algorithmic bias through learned latent structure. *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society* pp. 289–295 (2019)
3. Anderson, A.E., Foraker, A.G.: Centrilobular emphysema and panlobular emphysema: two different diseases 1. *Thorax* **28**, 547 – 550 (1973)
4. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: *Proceedings of the 37th International Conference on Machine Learning*. vol. 119, pp. 1597–1607 (2020)
5. Chen, T., Kornblith, S., Swersky, K., Norouzi, M., Hinton, G.E.: Big self-supervised models are strong semi-supervised learners. In: *NeurIPS*. vol. 33, pp. 22243–22255 (2020)
6. Choe, J., Lee, S.D., Do, K., Lee, G., Lee, J., Seo, J.: Deep learning-based image conversion of CT reconstruction kernels improves radiomics reproducibility for pulmonary nodules or masses. *Radiology* **292** **2**, 365–373 (2019)
7. Diwakar, M., Kumar, M.: A review on CT image noise and its denoising. *Biomed. Signal Process. Control*. **42**, 73–88 (2018)
8. Fortin, J., Sweeney, E., Muschelli, J., Crainiceanu, C., Shinohara, R.: Removing inter-subject technical variability in magnetic resonance imaging studies. *NeuroImage* **132**, 198–212 (2016)
9. Gretton, A., Fukumizu, K., Teo, C., Song, L., Schölkopf, B., Smola, A.: A kernel statistical test of independence. In: *NIPS*. vol. 20, pp. 585–592 (2007)
10. Grill, J.B., et al.: Bootstrap your own latent - a new approach to self-supervised learning. In: *NeurIPS*. vol. 33, pp. 21271–21284 (2020)
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. *IEEE CVPR* pp. 770–778 (2016)
12. Kim, B., Kim, H., Kim, K., Kim, S., Kim, J.: Learning not to learn: Training deep neural networks with biased data. *IEEE CVPR* pp. 9004–9012 (2019)
13. Kingma, D.P., Welling, M.: Auto-encoding variational Bayes. In: *2nd International Conference on Learning Representations, ICLR* (2014)
14. Krizhevsky, A.: Learning multiple layers of features from tiny images. *Tech. rep.*, University of Toronto (2009)
15. Leek, J., et al.: Tackling the widespread and critical impact of batch effects in high-throughput data. *Nat. Rev. Genet.* **11**, 733–739 (2010)
16. Mühlberg, A., et al.: The Technome - a predictive internal calibration approach for quantitative imaging biomarker research. *Scientific Reports* **10**(1103) (2020)
17. Shannon, C.: A mathematical theory of communication. *Bell Syst. Tech. J.* **27**, 379–423 (1948)
18. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **15**, 1929–1958 (2014)
19. Taubmann, O., Berger, M., Bögel, M., Xia, Y., Balda, M., Maier, A.: Computed tomography. In: Maier, A., Steidl, S., Christlein, V., Hornegger, J. (eds.) *Medical Imaging Systems – An Introductory Guide*, chap. 8, pp. 147–189. Springer International Publishing (2018)
20. Zhang, B.H., Lemoine, B., Mitchell, M.: Mitigating unwanted biases with adversarial learning. *AAAI/ACM Conf. on AI, Ethics, and Society* pp. 335–340 (2018)

DeepTechnome: Mitigating Unknown Bias in Deep Learning Based Assessment of CT Images (Supplementary Material)

Simon Langer¹, Oliver Taubmann², Felix Denzinger^{1,2}, Andreas Maier¹, and Alexander Mühlberg²

¹ Pattern Recognition Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

{simon.langer, felix.denzinger, andreas.maier}@fau.de

² Siemens Healthcare GmbH, Forchheim, Germany

oliver.taubmann@siemens-healthineers.com, alexander-muehlberg@hotmail.com



Fig. 1. We perform feasibility tests of our method on the well-known CIFAR-10 dataset (converted to grayscale), intentionally introducing technical variation and generating artificial control regions (CR) from the pixel-wise mean of all training images. During training, the network is exposed to dynamic additive white Gaussian noise (AWGN) on 90% of **dog** images; for testing a worst-case (adversarial) scenario, we apply dynamic AWGN to 90% of **cat** images. We also include static AWGN in our tests, with classes **truck** and **ship**, respectively. *From left to right:* 2 dynamic noise patterns, 2 static noise patterns, and 2 clean CRs only containing the minor background noise we always introduce. (Images taken directly from training input data, de-normalized back to $[0, 255]$).

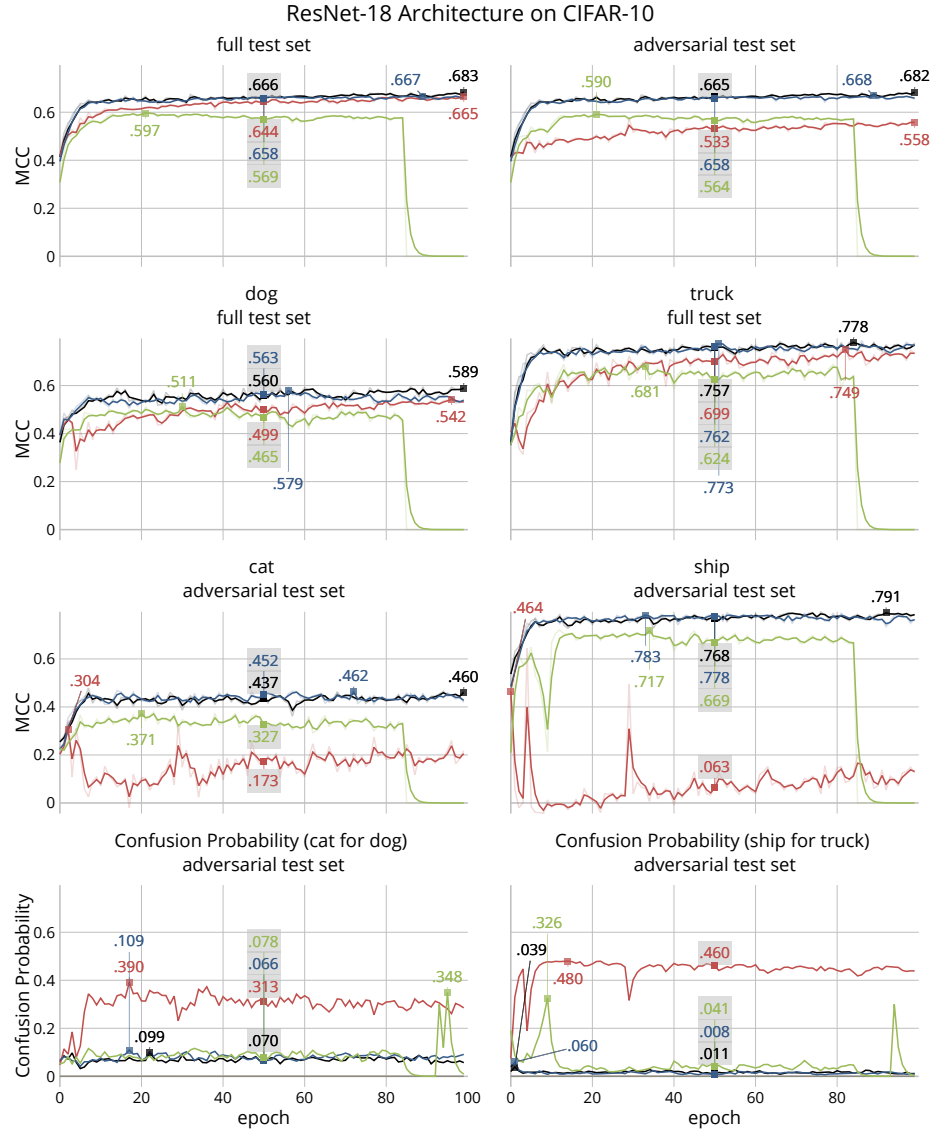


Fig. 2. Supplementary evaluation of *DecorreLayer* in ResNet-18, training on CIFAR-10. The top three rows visualize the test Matthews correlation coefficient (MCC) over the course of the training, displaying information on ROW 1) the entire test set, ROW 2) classes affected by noise at training time, ROW 3) classes adversarially manipulated with noise in our adversarial test set. The final ROW 4) displays a normalized (divided by 1000) entry of the confusion matrix at $y_i = \text{cat}/\text{ship}$ and $\hat{y}_i = \text{dog}/\text{truck}$. Training without *DecorreLayer* on biased data (■) yields a dramatic performance drop when technical variation is applied to different classes at test time (cat and ship adversarial test set), clearly caused by the model confusing cats for dogs and ships for trucks due to its reliance on technical variation. Applying our proposed *DecorreLayer* (■), we recover a majority of the performance on unbiased data (■), while not affecting training had no bias been present (■).

Table 1. Experimental Details – note that applying *DecorreLayer* did not noticeably increase training time (given in seconds per training epoch, noise of shared training server outweighs effects) or GPU memory consumption.

Architecture	Status <i>DecorreLayer</i>			
	◻/◻ → <i>DecorreLayer</i> off		◼/◼ → <i>DecorreLayer</i> on	
Small Custom	$7.3s \pm 1.1s$	2.2 GB	$7.5s \pm 1.0s$	2.3 GB
Medium Custom	$8.5s \pm 2.1s$	4.6 GB	$9.0s \pm 2.6s$	4.8 GB
ResNet-18	$11.9s \pm 1.7s$	2.7 GB	$10.4s \pm 1.6s$	2.8 GB











Table 2. Hyperparameters









	Parameter	Considered	Final
	Batch Size	[10, 100]	40
	Learning Rate	[0.00001, 0.5]	0.001, 0.02 (Med. Custom)
	Optimizer	{Adam, SGD}	SGD
<i>DecorreLayer</i> Filter Unit	Modes	{ <i>Factor</i> , <i>Sigmoid</i> , <i>Dropout</i> }	<i>Dropout</i>
	<i>Factor</i> constant c	[0, 0.5]	(unused) 0.01
	<i>Factor</i> threshold t	[0.05, 0.95]	(unused) 0.3
	<i>Sigmoid</i> middle m	0.3	(unused) 0.3
	<i>Sigmoid</i> steepness s	[5, 50000]	(unused) 50
	<i>Dropout</i> limit g	[0, 1.4]	0.3
Bias	Modes	{Kernel, Gaussian Noise}	—
	Attack Ratio	{0.7, 0.9}	—
	Gaussian Noise Std Dev	0.5 * Input Std Dev	—

Table 3. System specifications

OS	Ubuntu 18.04
Python Version	3.8
PyTorch Version	1.8
CPU	Intel Xeon Gold 6148 @ 2.40 GHz
GPU	Nvidia V100
RAM	756 GB

Table 4. Standard deviations in model performance (ROC-AUC) after 150 training epochs – supplements Table 1 in the main paper.

Train+Decorr			Small Custom Architecture							
Test	Decorr on  / 		S_1	S'_1	S_2	\tilde{S}_2	S_3	S'_3	S_4	\hat{S}_1
unbiased	unbiased		$\pm.025$	$\pm.025$	$\pm.025$	$\pm.025$	$\pm.025$	$\pm.025$	$\pm.025$	$\pm.038$
			$\pm.029$	$\pm.029$	$\pm.026$	$\pm.026$	$\pm.029$	$\pm.029$	$\pm.026$	$\pm.039$
	biased		$\pm.018$	$\pm.020$	$\pm.018$	$\pm.018$	$\pm.017$	$\pm.028$	$\pm.017$	$\pm.069$
			$\pm.043$	$\pm.024$	$\pm.041$	$\pm.045$	$\pm.048$	$\pm.028$	$\pm.051$	$\pm.096$
adversarial	unbiased		$\pm.026$	$\pm.026$	$\pm.026$	$\pm.026$	$\pm.026$	$\pm.026$	$\pm.026$	$\pm.041$
			$\pm.032$	$\pm.032$	$\pm.028$	$\pm.028$	$\pm.032$	$\pm.032$	$\pm.028$	$\pm.042$
	biased		$\pm.011$	$\pm.062$	$\pm.011$	$\pm.011$	$\pm.010$	$\pm.035$	$\pm.010$	$\pm.258$
			$\pm.042$	$\pm.019$	$\pm.097$	$\pm.107$	$\pm.102$	$\pm.029$	$\pm.094$	$\pm.053$

		Medium Custom Architecture							ResNet-18	
Test		M_1	M'_1	M_2	M_3	M'_3	M_4	\hat{M}_1	R_3	R_4
unbiased		$\pm.024$	$\pm.024$	$\pm.024$	$\pm.024$	$\pm.024$	$\pm.024$	$\pm.056$	$\pm.041$	$\pm.041$
		$\pm.036$	$\pm.036$	$\pm.029$	$\pm.036$	$\pm.036$	$\pm.029$	$\pm.040$	$\pm.036$	$\pm.060$
		$\pm.053$	$\pm.076$	$\pm.053$	$\pm.076$	$\pm.071$	$\pm.076$	$\pm.031$	$\pm.058$	$\pm.058$
		$\pm.019$	$\pm.054$	$\pm.041$	$\pm.083$	$\pm.037$	$\pm.030$	$\pm.079$	$\pm.035$	$\pm.051$
adversarial		$\pm.034$	$\pm.034$	$\pm.034$	$\pm.034$	$\pm.034$	$\pm.034$	$\pm.060$	$\pm.046$	$\pm.046$
		$\pm.037$	$\pm.037$	$\pm.037$	$\pm.037$	$\pm.037$	$\pm.037$	$\pm.041$	$\pm.043$	$\pm.067$
		$\pm.262$	$\pm.200$	$\pm.262$	$\pm.148$	$\pm.213$	$\pm.148$	$\pm.110$	$\pm.017$	$\pm.017$
		$\pm.018$	$\pm.066$	$\pm.262$	$\pm.085$	$\pm.042$	$\pm.035$	$\pm.227$	$\pm.048$	$\pm.069$