Learning De-biased Representations with Biased Representations

Hyojin Bahng [1] **Sanghyuk Chun** [2] **Sangdoo Yun** [2] **Jaegul Choo** [1] **Seong Joon Oh** [3]

Abstract

Many machine learning algorithms are trained and evaluated by splitting data from a single source into training and test sets. While such focus on *in-distribution* learning scenarios has led to interesting advancement, it has not been able to tell if models are relying on dataset biases as shortcuts for successful prediction (*e.g.*, using snow cues for recognising snowmobiles). Such biased models fail to generalise when the bias shifts to a different class. The *cross-bias generalisation* problem has been addressed by de-biasing training data through augmentation or re-sampling, which are often prohibitive due to the data collection cost (*e.g.*, collecting images of a snowmobile on a desert) and the difficulty of quantifying or expressing biases in the first place. In this work, we propose a novel framework to train a de-biased representation by encouraging it to be *different* from a set of representations that are biased by design. This tactic is feasible in many scenarios where it is much easier to define a set of biased representations than to define and quantify bias. We demonstrate the efficacy of our method across a variety of synthetic and real-world biases. Our experiments and analyses show that the method discourages models from taking bias shortcuts, resulting in improved generalisation.

1. Introduction

Most machine learning algorithms are trained and evaluated by randomly splitting a single source of data into training and test sets. Although this is a standard protocol, it is blind to a critical problem: the reliance on dataset bias (Torralba & Efros, 2011). For instance, many frog images are taken in swamp scenes, but swamp itself is not a frog. Nonetheless, a model will exploit this bias (*i.e.*, take "shortcuts") if it yields correct predictions for the majority of training examples. If the bias is sufficient to achieve high accuracy, there is little motivation for models to learn the complexity of the intended task, despite its full capacity to do so. Consequently, a model that relies on bias will achieve high in-distribution accuracy, yet fail to generalise when the bias shifts.

We tackle this "cross-bias generalisation" problem where a model does not exploit its full capacity due to the "sufficiency" of bias cues for prediction of the target label in the training data. For example, language models make predictions based on the presence of certain words (*e.g.*, "not" for "contradiction") (Gururangan et al., 2018) without much reasoning on the actual meaning of sentences, even if they are in principle capable of sophisticated reasoning. Similarly, convolutional neural networks (CNNs) achieve high accuracy on image classification by using local texture cues as shortcut, as opposed to more reliable global shape cues (Geirhos et al., 2019; Brendel & Bethge, 2019). 3D CNNs achieve high accuracy on video action recognition by relying on static cues as shortcut rather than capturing temporal actions (Weinzaepfel & Rogez, 2019; Li et al., 2018; Li & Vasconcelos, 2019).

Existing methods attempt to remove a model's dependency on bias by de-biasing the training data through augmentation (Geirhos et al., 2019) or introducing a pre-defined set of biases that a model is trained to be independent of (Wang et al., 2019a). Other approaches (Clark et al., 2019; Cadene et al., 2019) learn a biased model given source of bias as input, and de-bias through logit re-weighting or logit ensembling. These prior studies assume that biases can be easily defined or quantified, but often real-world biases do not (*e.g.*, texture or static bias above).

To address this limitation, we propose a novel framework to train a de-biased representation by encouraging it to be statistically independent from a set of representations that are biased by design. We use the Hilbert-Schmidt Independence Criterion (Gretton et al., 2005) to formulate the statistical independence as a regularisation term. Our insight is that there are certain types of bias that can be easily captured by defining a bias-characterising model (*e.g.*, CNNs of smaller receptive fields for texture bias; 2D CNNs for static bias in videos). Experiments show that our method is effective in reducing a model's dependency on "shortcuts" in training data, as evidenced by improved accuracy in test data where the bias is shifted or removed.

[1]Korea University [2]Clova AI Research, NAVER Corp. [3]Clova AI Research, LINE Plus Corp.. Correspondence to: Hyojin Bahng <hjbahng55@gmail.com>, Seong Joon Oh <coallaoh@gmail.com>.

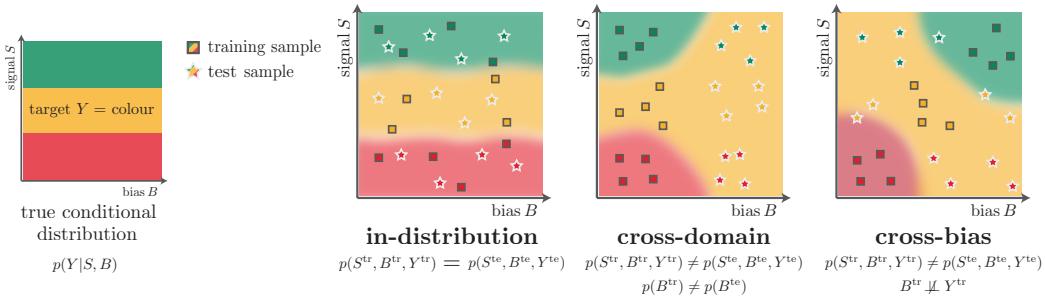


Figure 1. **Learning scenarios**. Different distributional gaps may take place between training and test distributions. Our work addresses the *cross-bias generalisation* problem. Background colours on the right three figures indicate the decision boundaries of models trained on given training data.

2. Problem Definition

We provide a rigorous definition of our over-arching goal: overcoming the bias in models trained on biased data. We systematically categorise the learning scenarios and cross-bias generalisation strategies.

2.1. Cross-bias generalisation

We first define random variables, signal S and bias B as cues for the recognition of an input X as certain target variable Y. Signals S are the cues essential for the recognition of X as Y; examples include the shape and skin patterns of frogs for frog image classification. Biases B's, on the other hand, are cues not essential for the recognition but correlated with the target Y; many frog images are taken in swamp scenes, so swamp scenes can be considered as B. A key property of B is that intervening on B should not change Y; moving a frog from swamp to a desert scene does not change the "frogness". We assume that the true predictive distribution $p(Y|X)$ factorises as $\int p(Y|S,B)p(S,B|X)$, signifying the sufficiency of $p(S,B|X)$ for recognition.

Under this framework, three learning scenarios are identified depending on the change of relationship $p(S,B,Y)$ across training and test distributions, $p(S^{tr},B^{tr},Y^{tr})$ and $p(S^{te},B^{te},Y^{te})$, respectively: in-distribution, cross-domain, and cross-bias generalisation. See Figure 1 for a summary.

In-distribution. $p(S^{tr},B^{tr},Y^{tr}) = p(S^{te},B^{te},Y^{te})$. This is the standard learning setup utilised in many benchmarks by splitting data from a single source into training and test data at random.

Cross-domain. $p(S^{tr},B^{tr},Y^{tr}) \neq p(S^{te},B^{te},Y^{te})$ and furthermore $p(B^{tr}) \neq p(B^{te})$. B in this case is often referred to as "domain". For example, training data consist of images with (Y^{tr}=frog, B^{tr}=wilderness) and (Y^{tr}=bird, B^{tr}=wilderness), while test data contain (Y^{te}=frog, B^{te}=indoors) and (Y^{te}=bird, B^{te}=indoors). This scenario is typically simulated by training and testing on different datasets (Ben-David et al., 2007).

Cross-bias. $p(B^{tr}) \not\perp\!\!\!\perp p(Y^{tr})$[1] and the dependency changes across training and test distributions: $p(B^{tr},Y^{tr}) \neq p(B^{te},Y^{te})$. We further assume that $p(B^{tr}) = p(B^{te})$, to clearly distinguish the scenario from the cross-domain generalisation. For example, training data only contain images of two types (Y^{tr}=frog, B^{tr}=swamp) and (Y^{tr}=bird, B^{tr}=sky), but test data contain unusual class-bias combinations (Y^{te}=frog, B^{te}=sky) and (Y^{te}=bird, B^{te}=swamp). Our work addresses this scenario.

2.2. Existing cross-bias generalisation methods and their assumptions

Under cross-bias generalisation scenarios, the dependency $p(B^{tr}) \not\perp\!\!\!\perp p(Y^{tr})$ makes bias B a viable cue for recognition. The model trained on such data becomes susceptible to interventions on B, limiting its generalisabililty when the bias is changed or removed in the test data. There exist prior approaches to this problem, but with different types and amounts of assumptions on B. We briefly recap the approaches based on the assumptions they require. In the next part §2.3, we will define our problem setting that requires an assumption distinct from the ones in prior approaches.

When an algorithm to disentangle bias B and signal S exists. Being able to disentangle B and S lets one collapse the feature space corresponding to B in both training and test data. A model trained on such normalised data then becomes free of biases. As ideal as it is, building a model to perfectly disentangle B and S is often unrealistic (*e.g.*, texture bias (Geirhos et al., 2019)). Thus, existing methods have proposed different approaches to tackle cross-bias generalisation.

When a data collection procedure or generative algorithm for $p(X|B)$ exists. When additional examples can be supplied through $p(X|B)$, the training dataset itself can be de-biased, *i.e.*, $B \perp\!\!\!\perp Y$. Such a data augmentation

[1] $\perp\!\!\!\perp$ and $\not\perp\!\!\!\perp$ denote independence and dependence, respectively.

strategy is indeed a valid solution adopted by many prior studies. Some approach has proposed to collect additional data to balance out the bias (Panda et al., 2018). Other approaches have proposed to synthesise data with a generative algorithm through image stylisation (Geirhos et al., 2019), object removal (Agarwal et al., 2019; Shetty et al., 2019), or generation of diverse, semantically similar linguistic variations (Shah et al., 2019; Ray et al., 2019). However, collecting unusual inputs can be expensive (Peyre et al., 2017), and building a generative model with pre-defined bias types (Geirhos et al., 2019) may suffer from bias misspecification or the lack of realism.

When a ground truth or predictive algorithm for $p(B|X)$ exists. Conversely, when one can tell the bias B for every input X, we can remove the dependency between the model predictions $f(X)$ and the bias B. The knowledge on $p(B|X)$ is provided in many realistic scenarios. For example, when the aim is to remove gender biases B in a job application process $p(Y|X)$, applicants' genders $p(B|X)$ are supplied as ground truths. Many existing approaches for fairness in machine learning have proposed independence-based regularisers to encourage $f(X) \perp\!\!\!\perp B$ (Zemel et al., 2013) or the conditional independence $f(X) \perp\!\!\!\perp B \,|\, Y$ (Quadrianto et al., 2019; Hardt et al., 2016). Other approaches have proposed to remove predictability of $p(B|X)$ based on $f(X)$ through domain adversarial losses (Wang et al., 2019b) or mutual information minimisation (Kim et al., 2019). When the ground truth of $p(B|X)$ is not provided, another approach has proposed to quantify texture bias by utilising the neural gray-level co-occurrence matrix and encouraging independence through projection (Wang et al., 2019a). However, there exist cases when B is difficult to even be defined or quantified but can only be indirectly specified.

2.3. Our scenario: Capturing bias with a set of models

Under the cross-bias generalisation scenario, some biases are not easily addressed by the above methods. Take texture bias as an example (§1, Geirhos et al. (2019)): (1) texture B and shape S cannot easily be disentangled, (2) collecting unusual images or building a generative model $p(X|B)$ is expensive, (3) building the predictive model $p(B|X)$ for texture requires enumeration (classifier) or embedding (regression) of all possible textures, which is not feasible.

However, slightly modifying the third assumption results in a problem setting that allows interesting application scenarios. Instead of assuming explicit knowledge on $p(B|X)$, we can approximate B by defining a set of models G that are biased towards B by design. For texture biases, for example, we define G to be the set of CNN architectures with small receptive fields. Then, any learned model $g \in G$ can by design make predictions $g(x)$ based on the patterns that can only be captured with small receptive fields (*i.e.*, textures), becoming more liable to overfit to texture.

More precisely, we define G to be a **bias-characterising model class** for the bias-signal pair (B, S) if for every possible joint distribution $p(B, X)$ there exists a $g \in G$ such that $p(B|X) \approx g(X)$ (**recall condition**) and every $g \in G$ satisfies $g(X) \perp\!\!\!\perp S \,|\, B$ (**precision condition**). In practice, G may not necessarily include all biases and may also capture important signals (*i.e.*, imperfect recall and precision). With this in mind, we formulate our framework as a regulariser to the original task so that $f(X)$ does not ignore every signal captured by G. We do not require G to be perfect.

There exist many scenarios when such G can be characterised, based on several empirical evidence for the type of bias. For instance, action recognition models rely heavily on static cues without learning temporal cues (Li et al., 2018; Li & Vasconcelos, 2019); we can regularise the 3D CNNs towards better generalisation across static biases by defining G to be the set of 2D CNNs. VQA models rely overly on language biases rather than visual cues (Agrawal et al., 2018). G can be defined as the set of models that only look at the language modality (Clark et al., 2019; Cadene et al., 2019). Entailment models are biased towards word overlap rather than understanding the underlying meaning of sentences (McCoy et al., 2019; Niven & Kao, 2019). We can design G to be the set of bag-of-words classifiers (Clark et al., 2019). Generally, these scenarios exemplify situations when the added architectural capacity is not fully utilised due to the sufficiency of simpler cues for solving the task in the given training set.

There are recent approaches that attempt to capture bias with bias-characterising models G and remove dependency on B via logit ensembling (Clark et al., 2019) or logit reweighting (Cadene et al., 2019). In §4, we empirically measure their performance on various types of synthetic and realistic biases.

3. Proposed Method

We present a solution for the cross-bias generalisation when the bias-characterising model class G is known (see §2.3); the method is referred to as `ReBias`. The solution consists of training a model f for the task $p(Y|X)$ with a regularisation term encouraging the independence between the prediction $f(X)$ and the set of all possible biased predictions $\{g(X) \,|\, g \in G\}$. We will introduce the precise definition of the regularisation term and discuss why and how it leads to the unbiased model.

3.1. `ReBias`: Removing bias with bias

If $p(B|X)$ is fully known, we can directly encourage $f(X) \perp\!\!\!\perp B$. Since we only have access to the set of bi-

ased models G (§2.3), we seek to promote $f(X) \perp\!\!\!\perp g(X)$ for every $g \in G$. Simply put, we de-bias a representation $f \in F$ by designing a set of biased models G and letting f run away from G. This leads to the independence from bias cues B while leaving signal cues S as valid recognition cues; see §2.3. We will specify `ReBias` learning objective after introducing our independence criterion, HSIC.

Hilbert-Schmidt Independence Criterion (HSIC). Since we need to measure the degree of independence between continuous random variables $f(X)$ and $g(X)$ in high-dimensional spaces, it is infeasible to resort to histogram-based measures; we use HSIC (Gretton et al., 2005). For two random variables U and V and kernels k and l, HSIC is defined as $\text{HSIC}^{k,l}(U,V) := ||C^{k,l}_{UV}||^2_{\text{HS}}$ where $C^{k,l}$ is the cross-covariance operator in the Reproducing Kernel Hilbert Spaces (RKHS) of k and l (Gretton et al., 2005), an RKHS analogue of covariance matrices. $||\cdot||_{\text{HS}}$ is the Hilbert-Schmidt norm, a Hilbert-space analogue of the Frobenius norm. It is known that for two random variables U and V and radial basis function (RBF) kernels k and l, $\text{HSIC}^{k,l}(U,V) = 0$ if and only if $U \perp\!\!\!\perp V$. A finite-sample estimate of $\text{HSIC}^{k,l}(U,V)$ has been used in practice for statistical testing (Gretton et al., 2005; 2008), feature similarity measurement (Kornblith et al., 2019), and model regularisation (Quadrianto et al., 2019; Zhang et al., 2018). We employ an unbiased estimator $\text{HSIC}^{k,l}_1(U,V)$ (Song et al., 2012) with m samples, defined as

$$\text{HSIC}^{k,l}_1(U,V) = \frac{1}{m(m-3)}\left[\text{tr}(\widetilde{U}\widetilde{V}^T) + \frac{\mathbf{1}^T\widetilde{U}\mathbf{1}\mathbf{1}^T\widetilde{V}^T\mathbf{1}}{(m-1)(m-2)} - \frac{2}{m-2}\mathbf{1}^T\widetilde{U}\widetilde{V}^T\mathbf{1}\right]$$

where $\widetilde{U}_{ij} = (1-\delta_{ij})\, k(u_i, u_j)$, $\{u_i\} \sim U$, *i.e.*, the diagonal entries of $\widetilde{U}$ are set to zero. $\widetilde{V}$ is defined similarly.

Minimax optimisation for bias removal. We define

$$\text{HSIC}^k_1(f(X), G(X)) := \max_{g \in G} \text{HSIC}^k_1(f(X), g(X)) \quad (1)$$

with an RBF kernel k for the degree of independence between representation $f \in F$ and the biased representations G. We write $\text{HSIC}_1(f,G)$ and $\text{HSIC}_1(f,g)$ as shorthands. The learning objective for f is then defined as

$$\min_{f \in F} \left\{ \mathcal{L}(f,X,Y) + \lambda \max_{g \in G} \text{HSIC}_1(f,g) \right\} \quad (2)$$

where $\mathcal{L}(f,X,Y)$ is the loss for the main task $p(Y|X)$ and $\lambda > 0$. We write $\mathcal{L}(f)$ as shorthands. Having specified G to represent the bias B, we need to train $g \in G$ for the original task to intentionally overfit G to B. Thus, the inner optimisation involves both the independence criterion and the original task loss $\mathcal{L}(g)$. The final learning objective for `ReBias` is then

$$\min_f \left\{ \mathcal{L}(f) + \lambda \max_g \Big(\text{HSIC}_1(f,g) - \lambda_g \mathcal{L}(g) \Big) \right\}. \quad (3)$$

$\lambda, \lambda_g > 0$. We solve equation 3 by alternative updates.

3.2. Why and how does it work?

Independence describes relationships between random variables, but we use it for function pairs. Which functional relationship does statistical independence translate to? In this part, we argue with proofs and observations that the answer to the above question is *the dissimilarity of invariance types learned by a pair of models*.

Linear case: Equivalence between independence and orthogonality. We study the set of function pairs (f,g) satisfying $f(X) \perp\!\!\!\perp g(X)$ for suitable random variable $X \sim p(X)$. Assuming linearity of involved functions and the normality of X, we obtain the equivalence between statistical independence and functional orthogonality.

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 $\ker(A)^\perp \perp_\Sigma \ker(B)^\perp$. For a positive semi-definite matrix Σ, we define $\langle r, s \rangle_\Sigma = \langle r, \Sigma s \rangle$, and the set orthogonality $\perp_\Sigma$ likewise. The proof is in Appendix.

In particular, when f and g have 1-dimensional outputs, the independence condition is translated to the orthogonality of their weight vectors and decision boundaries. From a machine learning point of view, f and g are models with orthogonal invariance types.

Non-linear case: HSIC as a metric learning objective. We lack theories to fully characterise general, possibly non-linear, function pairs (f,g) achieving $f(X) \perp\!\!\!\perp g(X)$; it is an interesting open question. For now, we make a set of observations in this general case, using the finite-sample independence criterion $\text{HSIC}_0(f,g) := (m-1)^{-2}\text{tr}(\widetilde{f}\,\widetilde{g}^T) = 0$, where $\widetilde{f}$ is the mean-subtracted kernel matrix $\widetilde{f}_{ij} = k(f(x_i), f(x_j)) - m^{-1}\sum_k k(f(x_i), f(x_k))$ and likewise for $\widetilde{g}$. Unlike in the loss formulation (§3.1), we use the biased HSIC statistic for simplicity.

Note that $\text{tr}(\widetilde{f}\,\widetilde{g}^T)$ is an inner product between flattened matrices $\widetilde{f}$ and $\widetilde{g}$. We consider the inner-product-minimising solution for f on an input pair $x_0 \neq x_1$ given a fixed g. The problem can be written as $\min_{f(x_0), f(x_1)} \ \text{tr}(\widetilde{f}\,\widetilde{g}^T)$, which is equivalent to $\min_{f(x_0), f(x_1)} \ \widetilde{f}_{01} \cdot \widetilde{g}_{10}$.

When $\widetilde{g}_{10} > 0$, g is relatively invariant on (x_1, x_0), since $k(g(x_1), g(x_0)) > m^{-1}\sum_i k(g(x_1), g(x_i))$. Then, the above problem boils down to $\min_{f(x_0), f(x_1)} \ \widetilde{f}_{01}$, signifying the relative *variance* of f on (x_0, x_1). Following a similar argument, we obtain the converse statement: if g is relatively variant on a pair of inputs, invariance of f on the pair minimises the objective.

We conclude that $\min_f \ \text{HSIC}_0(f,g)$ against a fixed g is

a metric-learning objective for the embedding f, where ground truth pairwise matches and mismatches are relative mismatches and matches for g, respectively. As a result, f and g learn different sorts of invariances.

Effect of HSIC regularisation on toy data. We have established that HSIC regularisation encourages the difference in model invariances. To see how it helps to de-bias a model, we have prepared synthetic two-dimensional training data following the cross-domain generalisation case in Figure 1: $X = (B, S) \in \mathbb{R}^2$ and $Y \in \{\text{red, yellow, green}\}$. Since the training data is perfectly biased, a multi-layer perceptron (MLP) trained on the data only shows 55% accuracy on de-biased test data (see decision boundary figure in Appendix). To overcome the bias, we have trained another MLP with equation 3 where the bias-characterising class G is defined as the set of MLPs that take only the bias dimension as input. This model exhibits de-biased decision boundaries (Appendix) with improved accuracy of 89% on the de-biased test data.

4. Experiments

In the previous section, `ReBias` has been introduced and theoretically justified. In this section, we present experimental results of `ReBias`. We first introduce the setup, including the biases tackled in the experiments, difficulties inherent to the cross-bias evaluation, and the implementation details (§4.1). Results on Biased MNIST (§4.2), ImageNet (§4.3) and action recognition (§4.4) are shown afterwards.

4.1. Experimental setup

Which biases do we tackle? Our work tackles the types of biases that arise due to the existence of shortcut cues that are sufficient for recognition in the training data. In the experiments, we tackle the "texture" biases for image classification and the "static" biases for video action recognition. Even if a CNN image classifier has wide receptive fields, empirical evidence indicates that they heavily rely on local texture cues for recognition, instead of the global shape cues (Geirhos et al., 2019). Similarly, a 3D CNN action recognition model that possesses the capacity to model temporal cues still rely on static cues like scenes or objects rather than the human motion for recognition (Weinzaepfel & Rogez, 2019). While it is difficult to precisely define and quantify all texture and scene types in the above examples, it is easy to intentionally design a model G to be biased to such cues. For the texture bias in image recognition, we design G as a CNN with smaller receptive fields; for the static bias in action recognition, we design G as a 2D CNN.

Evaluating cross-bias generalisation is difficult. To measure the performance of a model across real-world biases, one requires an unbiased dataset or one where the types and degrees of biases can be controlled. Unfortunately, data in real world arise with biases. To de-bias a frog and bird image dataset with swamp and sky (see §2.1), either rare data samples must be collected or one must generate such data; they are expensive procedures (Peyre et al., 2017).

We thus evaluate our method along two axes: (1) synthetic biases (Biased MNIST) and (2) realistic biases (ImageNet classification and action recognition task). Biased MNIST contains colour biases which we control in training and test data for an in-depth analysis of `ReBias`. For ImageNet classification, on the other hand, we use clustering-based proxy ground truths for texture bias to measure the cross-bias generalisability. For action recognition, we utilize the unbiased data that are publicly available (Mimetics), albeit in small quantity. We use the Mimetics dataset (Weinzaepfel & Rogez, 2019) for the unbiased test set accuracies, while using the biased Kinetics (Carreira & Zisserman, 2017) dataset for training. The set of experiments complement each other in terms of experimental control and realism.

Implementation of `ReBias`. We describe the specific design choices in `ReBias` implementation (equation 3). The source code is in the supplementary materials.

For texture biases, we define the biased model architecture families G as CNNs with small receptive fields (RFs). The biased models in G will by design learn to predict the target class of an image only through the local texture cues. On the other hand, we define a larger search space F with larger RFs for our unbiased representations.

In our work, all networks f and g are fully convolutional networks followed by a global average pooling (GAP) layer and a linear classifier. $f(x)$ and $g(x)$ denote the outputs of GAP layer (feature maps), on which we compute the independence measures using HSIC (§3.1).

For the Biased MNIST, F is a fully convolutional network with four convolutional layers with 7×7 kernels. Each convolutional layer uses the batch normalisation (Ioffe & Szegedy, 2015) and ReLU. G has the same architecture as F, except that the kernel sizes are 1×1. On ImageNet, we use the `ResNet18` (He et al., 2016) architecture for F with RF=435. G is defined as `BagNet18` (Brendel & Bethge, 2019) with RF=43. For action recognition, we use `3D-ResNet18` and `2D-ResNet18` for F and G whose RF along temporal dimension are 19 and 1, respectively.

We conduct experiments using the same batch size, learning rate, and epochs for fair comparison. We choose $\lambda = \lambda_g = 1$. For the Biased MNIST experiments, we set the kernel radius to one, while the median of distances is chosen for ImageNet and action recognition experiments. More implementation details are provided in Appendix.

	Biased						Unbiased					
ρ	Vanilla	Biased	HEX	LearnedMixin	RUBi	ReBias (ours)	Vanilla	Biased	HEX	LearnedMixin	RUBi	ReBias (ours)
.999	**100.**	**100.**	71.3	2.9	99.9	**100.**	10.4	10.	10.8	12.1	13.7	**22.7**
.997	**100.**	**100.**	77.7	6.7	99.4	**100.**	33.4	10.	16.6	50.2	43.0	**64.2**
.995	**100.**	**100.**	80.8	17.5	99.5	**100.**	72.1	10.	19.7	78.2	**90.4**	76.0
.990	**100.**	**100.**	66.6	33.6	**100.**	**100.**	89.1	10.	24.7	88.3	**93.6**	88.1
avg.	**100.**	**100.**	74.1	15.2	99.7	**100.**	51.2	10.	18.0	57.2	60.2	**62.7**

Table 1. **Biased MNIST results.** Biased and unbiased accuracies on varying train correlation ρ. Besides our results, we report vanilla F, G and previous methods. Ours are shown in gray columns. Each value is the average of three different runs.

Comparison methods. There are prior methodologies that can be applied in our cross-bias generalisation task (§2.2). We empirically compare ReBias against them. The prior methods include RUBi (Cadene et al., 2019) and LearnedMixin+H (Clark et al., 2019) that reduce the dependency of the model F on biases captured by G via logit re-weighting and logit ensembling, respectively. While the prior works additionally alter the training data for G, we only compare the objective functions themselves in our experiment. We additionally compare two methods that tackle texture bias: HEX (Wang et al., 2019a) and StylisedImageNet (Geirhos et al., 2019). HEX attempts to reduce the dependency of a model on "superficial statistics". It measures texture via neural grey-level co-occurrence matrices (NGLCM) and projects out the NGLCM feature from the model. StylisedImageNet reduces the models reliance on texture by augmenting the training data with texturised images.

4.2. Biased MNIST

We first verify our model on a dataset where we have full control over the type and amount of bias during training and evaluation. We describe the dataset and present the experimental results.

4.2.1. Dataset and evaluation

We construct a new dataset called **Biased MNIST** designed to measure the extent to which models generalise to bias shift. We modify MNIST (LeCun et al., 1998) by introducing the colour bias that highly correlate with the label Y during training. With B alone, a CNN can achieve high accuracy without having to learn inherent signals for digit recognition S, such as shape, providing little motivation for the model to learn beyond these superficial cues.

Figure 2. **Biased MNIST.** A synthetic dataset with the colour bias which highly correlates with the labels during training.

We inject the colour bias by adding a colour on training image backgrounds (Figure 2). We pre-select 10 distinct colours for each digit $y \in \{0, \cdots, 9\}$. Then, for each image of digit y, we assign the pre-defined colour $b(y)$ with probability $\rho \in [0, 1]$ and any other colour with probability $(1-\rho)$. ρ then controls the bias-target correlation in the training data: $\rho = 1.0$ leads to complete bias and $\rho = 0.1$ leads to an unbiased dataset. We consider $\rho \in \{0.99, 0.995, 0.997, 0.999\}$ to simulate significant amounts of bias during training. We evaluate the models generalisability to bias shift by evaluating under the following criterion:

Biased. $p(S^{\text{te}}, B^{\text{te}}, Y^{\text{te}}) = p(S^{\text{tr}}, B^{\text{tr}}, Y^{\text{tr}})$, the in-distribution case in §2.1. Whatever bias the training set contains, it is replicated in the test set (same ρ). This measures the ability of de-biased models to maintain high in-distribution performances while generalising to unbiased settings.

Unbiased. $B^{\text{te}} \perp\!\!\!\perp Y^{\text{te}}$. We assign biases on test images independently of the labels. Bias is no longer predictive of Y and a model needs to utilise actual signals S to yield correct predictions.

4.2.2. Results

Results on the Biased MNIST are shown in Table 1.

ReBias lets a model overcome bias. We observe that vanilla F achieves 100% accuracy under the "biased" metric (the same bias between training and test data) in the Biased MNIST for all ρ. This is how most machine learning tasks are evaluated, yet this does not show the extent to which the model depends on bias for prediction. When the bias cues are randomly assigned to the label at evaluation, vanilla F accuracy collapses to 10.4% under the "unbiased" metric on the Biased MNIST when the train correlation is large, *i.e.*, $\rho = 0.999$. The intentionally biased models G result in 10.0% on the Biased MNIST, the random chance performance, for all ρ. This exemplifies the case where a seemingly high-performing model has in fact overfitted to bias and does not generalise to new situations.

On the other hand, ReBias achieves robust generalisation across all settings by learning to be different from the representations G. Especially, ReBias achieves better unbiased accuracies than the vanilla model under the highly correlated settings: $10.4 \rightarrow 22.7\%$ and $33.4 \rightarrow 64.2\%$ boosts for ρ =0.999 and 0.997, respectively.

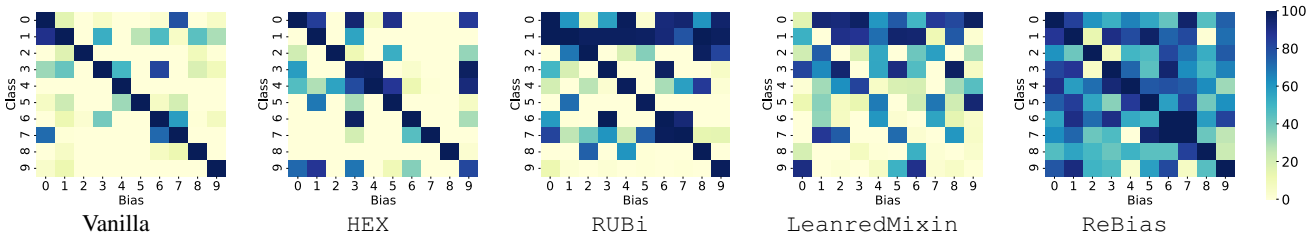


Figure 3. **Accuracy per bias-class pair.** We show accuracies for each bias and class pair $(B, Y) = (b, y)$ on Biased MNIST. All methods are trained with $\rho = 0.997$. The diagonals in each matrix indicate the pre-defined bias-target pair (§4.2.1). The number of samples per (b, y) cell is identical across all pairs (unbiased test set).

Comparison against other methods. As `HEX` predefines bias as patterns captured by NGLCM, we observe that it does not improve generalisability to colour bias (18.0%) while also hurting the in-distribution accuracy (74.1%) compared to vanilla F. `LearnedMixin` achieves performance gain in unbiased accuracies (57.2%) yet suffers a severe performance drop for unbiased accuracies (15.2%). `RUBi` achieves robust generalisation across biased and unbiased accuracies (99.7% and 60.2% respectively). We show in the following experiments that `LearnedMixin` and `RUBi` achieve sub-optimal performances in realistic texture and static biases.

Analysis of per-bias performances. In Figure 3, we provide more fine-grained results by visualising the accuracies per bias-class pair $(B, Y) = (b, y)$. The diagonal average corresponds to the *biased accuracy* and the overall average corresponds to the *unbiased accuracy*. We observe that the vanilla model has higher accuracies on diagonals and lower on off-diagonals, showing the heavy reliance on the colour (bias) cues. `HEX` and `RUBi` demonstrate sporadic improvements in certain off-diagonal cells, but the overall improvements are limited. `LearnedMixin` shows further enhancements, yet it shows near-zero accuracies on diagonal entries (also seen in Table 1). `ReBias` uniformly improves the off-diagonal accuracies, while not sacrificing the diagonal accuracies.

Learning Curves. In Figure 4, we plot the evolution of unbiased accuracy and HSIC values as `ReBias` is trained. `ReBias` is trained with $\rho = 0.997$ and tested with $\rho = 0.1$ (unbiased). While the classification loss alone, *i.e.*, vanilla F, leads to an unbiased accuracy of 33.4%, the unbiased accuracy increases dramatically ($> 60\%$) as the HSIC between F and G is minimised during training. We observe that there exists a strong correlation between the HSIC values and unbiased accuracies.

4.3. ImageNet

In ImageNet experiments, we further validate the applicability of `ReBias` on the texture bias in realistic images

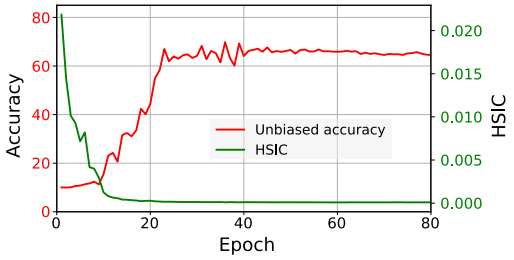


Figure 4. **Learning curves.** `ReBias` achieves better generalisation by minimizing HSIC between representations.

(*i.e.*, objects in natural scenes). The texture bias often lets a model achieve good in-distribution performances by exploiting the local texture shortcuts (*e.g.*, determining a swan class by not seeing its shape but the background water texture).

4.3.1. Dataset and Evaluation

We construct **9-Class ImageNet**, a subset of ImageNet (Russakovsky et al., 2015) containing 9 super-classes as done in Ilyas et al. (2019), since a full-scale analysis on ImageNet is not scalable. We additionally balance the ratios of subclass images for each super-class to focus on the effect of texture bias.

Since it is difficult to evaluate the cross-bias generalisability on realistic unbiased data (§4.1), we settle for alternative evaluations:

Biased. $p(S^{\text{te}}, B^{\text{te}}, Y^{\text{te}}) = p(S^{\text{tr}}, B^{\text{tr}}, Y^{\text{tr}})$. Accuracy is measured on the in-distribution validation set. Though widely-used, this metric is blind to a model's generalisability to unseen bias-target combinations.

Unbiased. $B^{\text{te}} \perp\!\!\!\perp Y^{\text{te}}$. As a proxy to the perfectly debiased test data, which is difficult to collect (§4.1), we use *texture* clusters IDs $c \in \{1, \cdots, K\}$ as the ground truth labels for texture bias obtained by the k-means clustering. For full details of texture clustering algorithm, see Appendix. For an unbiased accuracy measurement, we compute the accuracies for every set of images corresponding to a texture-class combination (c, y). The combination-wise

accuracy $A_{c,y}$ is computed by $\text{Corr}(c,y)/\text{Pop}(c,y)$, where $\text{Corr}(c,y)$ is the number of correctly predicted samples in (c,y) and $\text{Pop}(c,y)$ is the total number of samples in (c,y), called the **population** at (c,y). The unbiased accuracy is then the mean accuracy over all $A_{c,y}$ where the population $\text{Pop}(c,y) > 10$. This measure gives more weights on samples of unusual texture-class combinations (smaller $\text{Pop}(c,y)$) that are less represented in the usual biased accuracies. Under this unbiased metric, a biased model basing its recognition on textures is likely to show sub-optimal results on unusual combinations, leading to a drop in the unbiased accuracy. Since the k-means clustering is non-convex, we report the average unbiased accuracy of three clustering results with different initial points.

ImageNet-A. ImageNet-A (Hendrycks et al., 2019) contains the failure cases of ImageNet-trained `ResNet50` among web images. The images consist of many failure modes of networks when "frequently appearing background elements" (Hendrycks et al., 2019) become erroneous cues for recognition (*e.g.* a bee image feeding on hummingbird feeder is recognised as a hummingbird). An improved performance on ImageNet-A is an indirect signal that the model learns beyond the bias shortcuts.

4.3.2. Results

We measure performances of `ResNet18` trained under the `ReBias` framework to be different from `BagNet18`. We use the metrics in the previous part. Results are shown in Table 2.

Vanilla models are biased. `ResNet18` shows good performances on the biased accuracy (90.8%) but dropped performances on the texture-unbiased accuracy (88.8%). `BagNet18` performs worse than the vanilla ResNet as they are heavily biased towards texture by design (*i.e.*, small receptive field sizes). The drop signifies the biases of vanilla models towards texture cues; by basing their predictions on texture cues they obtain generally better accuracies on texture-class pairs (c,y) that are more represented. The drop also shows the limitation of current evaluation schemes where the cross-bias generalisation is not measured.

`ReBias` leads to less biased models. When `ReBias` is applied on `ResNet18` to make it learn cues beyond those captured by `BagNet18`, we observe a general boost in the biased, unbiased, and ImageNet-A accuracies (Table 2). The unbiased accuracy of `ResNet18` improves from 88.8% to 90.5%, thus robustly generalising to less represented texture-class combinations at test time. Our method also shows improvements on the challenging ImageNet-A subset (*e.g.* from 24.9% to 29.6%), which further shows an improved generalisation. While `StylisedImageNet` attempts to mitigate texture bias by stylisation, it does not increase the generalisability for both the unbiased and ImageNet-A accuracy (86.6% and 24.6% respectively). Similar to the Biased MNIST results, `Learned-Mixin` suffers a collapse in the in-distribution accuracy (from 90.8% to 67.9%) and does not improve generalisability to less represented texture-class combinations or the challenging ImageNet-A. `RUBi` only shows improvement on ImageNet-A (from 24.9% to 27.7%).

Model description	Biased	Unbiased	IN-A
Vanilla (`ResNet18`)	90.8	88.8	24.9
Biased (`BagNet18`)	67.7	65.9	18.8
`StylisedIN` (Geirhos et al., 2019)	88.4	86.6	24.6
`LearnedMixin` (Clark et al., 2019)	64.1	62.7	15.0
`RUBi` (Cadene et al., 2019)	90.5	88.6	27.7
`ReBias` (ours)	**91.9**	**90.5**	**29.6**

Table 2. **ImageNet results.** We show results corresponding to F =`ResNet18` and G =`BagNet18`. IN-A indicates ImageNet-A. We repeat each experiment three times.

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

Table 3. **Action recognition results.** We show results corresponding to F =`3D-ResNet18` and G =`2D-ResNet18` with baseline comparisons. Top-1 accuracies are reported. Each result is the average of three runs.

4.4. Action recognition

To see further effectiveness of `ReBias` on reducing static biases in a video understanding task, we conduct the action recognition experiments with 3D CNNs. 3D CNNs have proven their state-of-the-art performances on action recognition benchmarks such as Kinetics (Carreira & Zisserman, 2017), but recent studies (Sevilla-Lara et al., 2019; Li et al., 2018; Li & Vasconcelos, 2019) have shown that such action datasets have strong static biases towards the scene or objects in videos. As a result, 3D CNNs make predictions dominantly based on static cues, despite their ability to capture temporal signals, and they achieve high accuracies even with temporal cues removed (*e.g.*, shuffling frames or masking-out human actor in videos) (Weinzaepfel & Rogez, 2019). This bias problem occasionally leads to performance drop when static cues shift across training and test settings (*e.g.*, predicting "swimming" class when a person plays football near a swimming pool).

4.4.1. Dataset

We use the Kinetics dataset (Carreira & Zisserman, 2017) for training, which is known to have bias towards static cues. To evaluate the cross-bias generalisability, we use the

Mimetics dataset (Weinzaepfel & Rogez, 2019) that consists of videos of a mime artist performing actions without any context. The classes of Mimetics are fully covered by the Kinetics classes and we use it as the unbiased validation set. Since the training and testing of the full action datasets are not scalable, we sub-sample 10-classes from both datasets. Detailed dataset descriptions are in Appendix.

4.4.2. RESULTS

We evaluate the performances of `3D-ResNet18` trained to be different from the biased model `2D-ResNet18`. Main results are shown in Table 3.

Vanilla model is biased. The vanilla `3D-ResNet18` model F shows a reasonable performance on the biased Kinetics with 54.5% accuracy, but significantly loses the accuracy on the unbiased Mimetics with 18.9% accuracy. While `3D-ResNet18` is originally designed for capturing temporal signals within videos, it relies a lot on static cues, resulting in a similar performance with the 18.4% accuracy by `2D-ResNet18` on Mimetics.

`ReBias` reduces the static bias. Applying `ReBias` on `3D-ResNet18` encourages it to utilise the temporal modelling capacity by forcing it to reason differently from `2D-ResNet18`. `ReBias` improves the accuracies on both Kinetics and Mimetics datasets beyond the vanilla model F: $54.5 \rightarrow 55.8\%$ and $18.9 \rightarrow 22.4\%$, respectively. We also compare against the two baseline methods, `LearnedMixin` (Clark et al., 2019) and `RUBi` (Cadene et al., 2019), as in the previous sections. `ReBias` shows better performances than the two baseline methods for reducing the static bias for action recognition. We believe that the difficulty of the action recognition task on Kinetics hampers the normal operation of the logit-modification step in the baseline methods, severely hindering the convergence with respect to the cross-entropy loss. The training of `ReBias`, on the other hand, remains stable as the independence loss acts only as a regularisation term.

5. Conclusion

We have identified a practical problem faced by many machine learning algorithms that the learned models exploit bias shortcuts to recognise the target: the cross-bias generalisation problem (§2). Models tend to under-utilise its capacity to extract non-bias signals (*e.g.* global shapes for object recognition, or temporal actions for action recognition) when bias shortcuts provide sufficient cues for recognition in the training data (*e.g.* texture for object recognition, or static contexts for action recognition) (Geirhos et al., 2019; Weinzaepfel & Rogez, 2019). We have addressed this problem with the `ReBias` method. Given an identified set of models G that encodes the bias to be removed, `ReBias` encourages a model f to be statistically independent of G (§3). We have provided theoretical justifications (§3.2) and have validated the superiority of `ReBias` in removing biases from models through experiments on Biased MNIST, ImageNet classification, and the Mimetics action recognition benchmark (§4).

Acknowledgements

We thank Clova AI Research team for the discussion and advice, especially Dongyoon Han, Youngjung Uh, Yunjey Choi, Byeongho Heo, Junsuk Choe, Muhammad Ferjad Naeem, and Hyojin Park for their internal reviews. Naver Smart Machine Learning (NSML) platform (Kim et al., 2018) has been used in the experiments. Kay Choi has helped the design of Figure 1.

References

Agarwal, V., Shetty, R., and Fritz, M. Towards causal vqa: Revealing and reducing spurious correlations by invariant and covariant semantic editing. *arXiv preprint arXiv:1912.07538*, 2019.

Agrawal, A., Batra, D., Parikh, D., and Kembhavi, A. Don't just assume; look and answer: Overcoming priors for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4971–4980, 2018.

Ben-David, S., Blitzer, J., Crammer, K., and Pereira, F. Analysis of representations for domain adaptation. In *Advances in neural information processing systems*, pp. 137–144, 2007.

Brendel, W. and Bethge, M. Approximating CNNs with bag-of-local-features models works surprisingly well on imagenet. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SkfMWhAqYQ.

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.

Gretton, A., Bousquet, O., Smola, A., and Schölkopf, B. Measuring statistical dependence with hilbert-schmidt norms. In *International conference on algorithmic learning theory*, pp. 63–77. Springer, 2005.

Gretton, A., Fukumizu, K., Teo, C. H., Song, L., Schölkopf, B., and Smola, A. J. A kernel statistical test of independence. In *Advances in neural information processing systems*, pp. 585–592, 2008.

Gururangan, S., Swayamdipta, S., Levy, O., Schwartz, R., Bowman, S., and Smith, N. A. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 107–112, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/N18-2017.

Hardt, M., Price, E., Srebro, N., et al. Equality of opportunity in supervised learning. In *Advances in neural information processing systems*, pp. 3315–3323, 2016.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hendrycks, D., Zhao, K., Basart, S., Steinhardt, J., and Song, D. Natural adversarial examples. *arXiv preprint arXiv:1907.07174*, 2019.

Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., and Madry, A. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems*, pp. 125–136, 2019.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Bach, F. and Blei, D. (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 448–456, Lille, France, 07–09 Jul 2015. PMLR.

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.

Kim, B., Kim, H., Kim, K., Kim, S., and Kim, J. Learning not to learn: Training deep neural networks with biased data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9012–9020, 2019.

Kim, H., Kim, M., Seo, D., Kim, J., Park, H., Park, S., Jo, H., Kim, K., Yang, Y., Kim, Y., et al. Nsml: Meet the mlaas platform with a real-world case study. *arXiv preprint arXiv:1810.09957*, 2018.

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

Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, 2019.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Li, Y. and Vasconcelos, N. Repair: Removing representation bias by dataset resampling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9572–9581, 2019.

Li, Y., Li, Y., and Vasconcelos, N. Resound: Towards action recognition without representation bias. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 513–528, 2018.

McCoy, R. T., Pavlick, E., and Linzen, T. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Niven, T. and Kao, H.-Y. Probing neural network comprehension of natural language arguments. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Panda, R., Zhang, J., Li, H., Lee, J.-Y., Lu, X., and Roy-Chowdhury, A. K. Contemplating visual emotions: Understanding and overcoming dataset bias. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 579–595, 2018.

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.

Peyre, J., Sivic, J., Laptev, I., and Schmid, C. Weakly-supervised learning of visual relations. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5179–5188, 2017.

Quadrianto, N., Sharmanska, V., and Thomas, O. Discovering fair representations in the data domain. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8227–8236, 2019.

Ray, A., Sikka, K., Divakaran, A., Lee, S., and Burachas, G. Sunny and dark outside?! improving answer consistency in vqa through entailed question generation. 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. 5860–5865, 2019.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Sevilla-Lara, L., Zha, S., Yan, Z., Goswami, V., Feiszli, M., and Torresani, L. Only time can tell: Discovering temporal data for temporal modeling. *arXiv preprint arXiv:1907.08340*, 2019.

Shah, M., Chen, X., Rohrbach, M., and Parikh, D. Cycle-consistency for robust visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6649–6658, 2019.

Shetty, R., Schiele, B., and Fritz, M. Not using the car to see the sidewalk–quantifying and controlling the effects of context in classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8218–8226, 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.

Torralba, A. and Efros, A. A. Unbiased look at dataset bias. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1521–1528, June 2011.

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.

Wang, H., He, Z., and Xing, E. P. Learning robust representations by projecting superficial statistics out. In *International Conference on Learning Representations*, 2019a. URL https://openreview.net/forum?id=rJEjjoR9K7.

Wang, T., Zhao, J., Yatskar, M., Chang, K.-W., and Ordonez, V. Balanced datasets are not enough: Estimating and mitigating gender bias in deep image representations. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5310–5319, 2019b.

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

Zemel, R., Wu, Y., Swersky, K., Pitassi, T., and Dwork, C. Learning fair representations. In *International Conference on Machine Learning*, pp. 325–333, 2013.

Zhang, C., Liu, Y., Liu, Y., Hu, Q., Liu, X., and Zhu, P. Fish-mml: Fisher-hsic multi-view metric learning. In *International Joint Conference on Artificial Intelligence*, pp. 3054–3060, 2018.

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 $\arg\min_{g \in G} \lambda_g \, \mathcal{L}(g, x, y) - \mathrm{HSIC}_1(f(x), g(x))$.
  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

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 A1. **Network architecture for action recognisers.** The kernel dimensions of convolution layers are denoted as [T × W × H, 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.

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).

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 A1.

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 A2 (Biased MNIST), Table A3 (ImageNet) and Table A4 (action recognition).

E. Decision Boundary Visualisation for Toy Experiment

We show the decision boundaries of the toy experiment (§3.2) in Figure A1. GIF animations are at anonymous link.

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 A2. **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 A3. **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 A4. **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.

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

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).

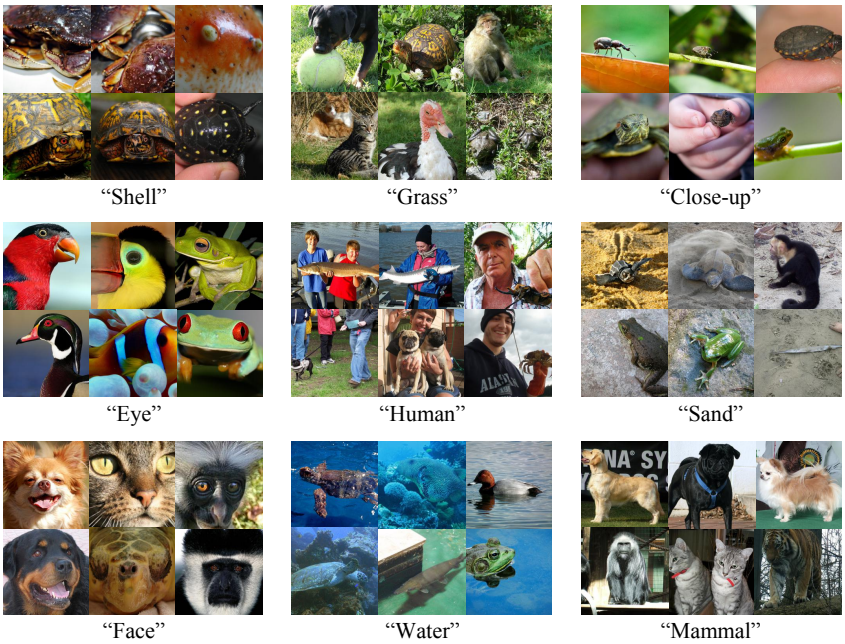
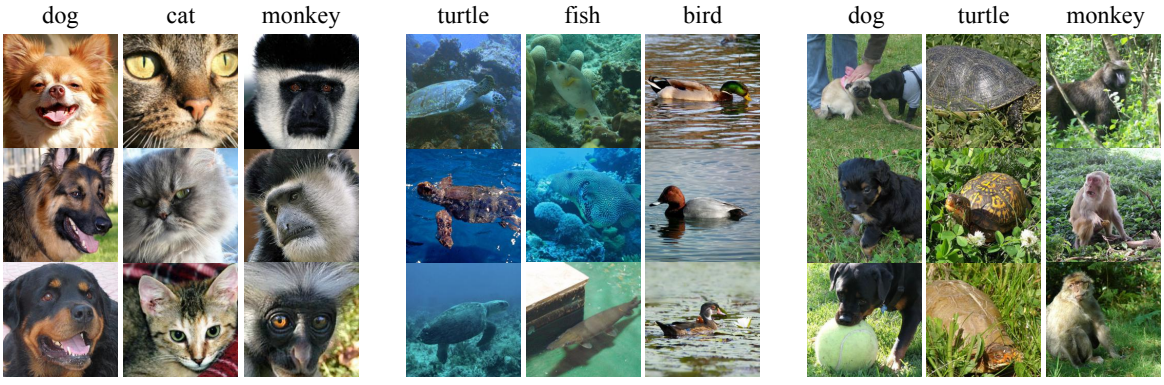


Figure A2. **Texture clusters.** Example images from texture clusters. Each cluster is named according to the common texture pattern.



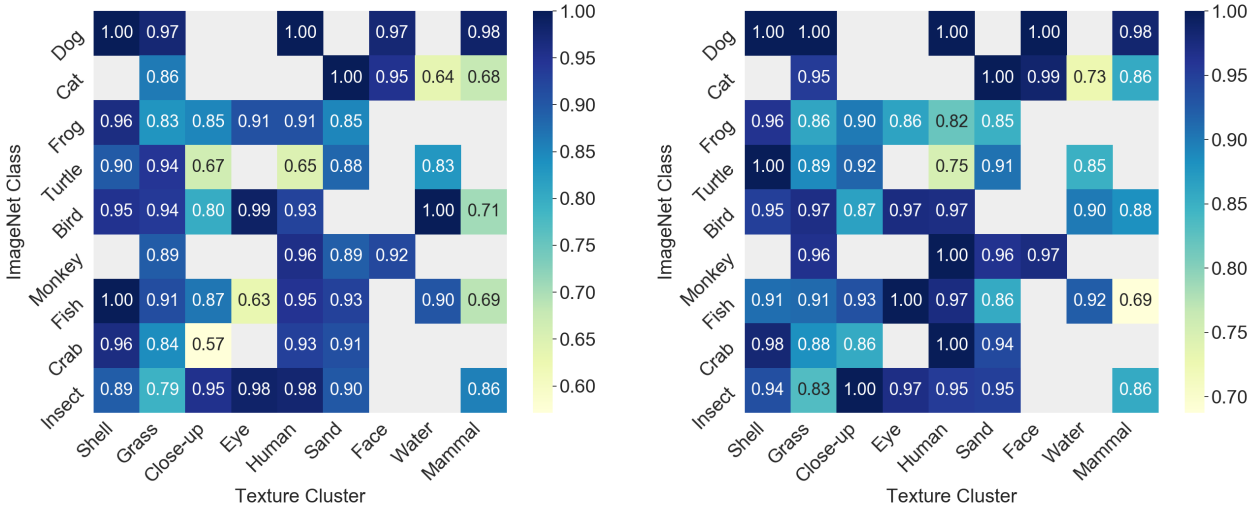
(a) "Face" cluster samples (b) "Water" cluster samples (c) "Grass" cluster samples

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 present in the dataset, we count the number of samples for each texture-class pair (c, y) (denoted "population" $\text{Pop}(c, y)$ in the main paper). For each class, we define the *dominant texture cluster* as the largest cluster that contains more than 30% of the class samples. 4 out of 9 classes considered has 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).

(a) Canoeing or kayaking

(b) Surfing water

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.