# BiasBed – Rigorous Texture Bias Evaluation

Nikolai Kalischek[1] Rodrigo Caye Daudt[1] Torben Peters[1]
Reinhard Furrer[2] Jan D. Wegner[3] Konrad Schindler[1]
[1]Photogrammetry and Remote Sensing, ETH Zürich
[2]Institute for Mathematics, University of Zurich
[3]Institute for Computational Science, University of Zurich

## Abstract

*The well-documented presence of texture bias in modern convolutional neural networks has led to a plethora of algorithms that promote an emphasis on shape cues, often to support generalization to new domains. Yet, common datasets, benchmarks and general model selection strategies are missing, and there is no agreed, rigorous evaluation protocol. In this paper, we investigate difficulties and limitations when training networks with reduced texture bias. In particular, we also show that proper evaluation and meaningful comparisons between methods are not trivial. We introduce* BiasBed*, a testbed for texture- and style-biased training, including multiple datasets and a range of existing algorithms. It comes with an extensive evaluation protocol that includes rigorous hypothesis testing to gauge the significance of the results, despite the considerable training instability of some style bias methods. Our extensive experiments, shed new light on the need for careful, statistically founded evaluation protocols for style bias (and beyond).* E.g.*, we find that some algorithms proposed in the literature do not significantly mitigate the impact of style bias at all. With the release of* BiasBed*, we hope to foster a common understanding of consistent and meaningful comparisons, and consequently faster progress towards learning methods free of texture bias. Code is available at* https://github.com/D1noFuzi/BiasBed

## 1. Introduction

Visual object recognition is fundamental to our daily lives. Identifying and categorizing objects in the environment is essential for human survival, indeed our brain is able to assign the correct object class within a fraction of a second, independent of substantial variations in appearance, illumination and occlusion [28]. Recognition mainly takes place along the ventral pathway [7], i.e., visual perception induces a hierarchical processing stream from local patterns to complex features, in feed-forward fashion [28]. Signals are filtered to low frequency and used in parallel also in a top-down manner [2], emphasizing the robustness and invariance to deviations in appearance.

Inspired by our brain's visual perception, convolutional neural architectures build upon the hierarchical intuition, stacking multiple convolutional layers to induce feed-forward learning of basic concepts to compositions of complex objects. Indeed, early findings suggested that neurons in deeper layers are activated by increasingly complex shapes, while the first layers are mainly tailored towards low-level features such as color, texture and basic geometry. However, recent work indicates the opposite: convolutional neural networks (CNNs) exhibit a strong bias to base their decision on texture cues [11–13, 17], which heavily influences their performance, in particular under domain shifts, which typically affect local texture more than global shape.

This inherent texture bias has led to a considerable body of work that tries to minimize texture and style bias and shift towards "human-like" shape bias in object recognition. Common to all approaches is the principle of incorporating adversarial texture cues into the learning pipeline – either directly in input space [13, 15, 27] or implicitly in feature space [20,22,30,33]. Perturbing the texture cues in the input forces the neural network to make "texture-free" decisions, thus focusing on the objects' shapes that remain stable during training. While texture cues certainly boost performance in fine-grained classification tasks, where local texture patterns may indicate different classes, they can cause serious harm when the test data exhibit a domain shift w.r.t. training. In this light, texture bias can be seen as a main reason for degrading domain generalization performance, and various algorithms have been developed to improve generalization to new domains with different texture properties (respectively image styles), *e.g.*, [13, 15, 20, 22, 27, 30, 33, 40].

However, while a considerable number of algorithms have been proposed to address texture bias, they are neither evaluated on common datasets nor with common evaluation metrics. Moreover they often employ inconsistent model selection criteria or do not report at all how model selection

is done. With the present paper we promote the view that:

- the large domain shifts induced by texture-biased training cause large fluctuations in accuracy, which call for a particularly rigorous evaluation;
- model selection has so far been ignored in the literature; together with the high training volatility, this may have lead to overly optimistic conclusions from spurious results;
- in light of the difficulties of operating under drastic domain shifts, experimental results should include a notion of uncertainty in the evaluation metrics.

Motivated by these findings, we have implemented *BiasBed*, an open-source PyTorch [35] testbed that comes with six datasets of different texture and shape biases, four adversarial robustness datasets and eight fully implemented algorithms. Our framework allows the addition of new algorithms and datasets with few lines of code, including full flexibility of all parameter settings. In order to tackle the previously discussed limitations and difficulties for evaluating such algorithms, we have added a carefully designed evaluation pipeline that includes training multiple runs and employing multiple model selection methods, and we report all results based on sound statistical hypothesis tests – all run with a single command. In addition to our novel framework, the present paper makes the following contributions:

- We highlight shortcomings in the evaluation protocols used in recent work on style bias, including the observation that there is a very high variance in the performance of different runs of the same algorithm, and even between different checkpoints in the same run with similar validation scores.
- We develop and openly release a testbed that rigorously compares different algorithms using well-established hypothesis testing methods. This testbed includes several of the most prominent algorithms and datasets in the field, and is easily extensible.
- We observe in our results that current algorithms on texture-bias datasets fail to surpass simple ERM in a statistically significant way, which is the main motivation for this work and for using rigorous hypothesis tests for evaluating style bias algorithms.

In Sec. 2, we provide a comprehensive overview of existing work on reducing texture bias. Furthermore we describe the main forms of statistical hypothesis testing. We continue in Sec. 3 with a formal introduction to biased learning, and systematically group existing algorithms into families with common properties. In Sec. 4, we investigate previous evaluation practices in detail, discuss their limitations, and give a formal introduction to hypothesis testing. Sec. 5 describes our proposed *BiasBed* evaluation framework in detail, while Sec. 6 presents experiments, followed by a discussion (Sec. 7), conclusions (Sec. 8) and a view towards the broader impact of our work (Sec. 9).

## 2. Related work

**Texture and style bias.** The seminal work of Geirhos *et al*. [13] showed that modern neural networks are heavily biased towards local patterns, commonly referred to as texture or style.[1] The finding that, contrary to earlier hypotheses [23, 24], the network output is dominated by domain-specific texture rather than generic, large-scale shape cues helps to explain their poor ability to generalize to unseen image domains. As a consequence, several authors have tried to address the issue under the umbrella term *Domain Generalization* [20, 22, 30], although *texture debiasing* would be a more accurate description of their approach: they take advantage of neural style transfer ideas [11, 21] and alter the texture features, thereby forcing the model to rely more on shape features. To that end, Geirhos *et al*. [13] put forward a new *stylized* ImageNet dataset. That dataset contains images that are blended with a random texture image to generate a diverse set of "texture-independent" samples, such that the contained texture is not informative about the class label anymore. However, enforcing shape bias tends to deteriorate performance on the source domain, *i.e*. the domain trained on, *e.g*. ImageNet. To balance in-domain and out-of-domain (OOD) performance, Li *et al*. [27] propose debiased learning by linearly interpolating the target labels between the texture class and the content class.

Inspired by neural style transfer, a whole palette of work [20, 22, 30, 33, 40] deals with changing the texture statistics in feature space. The authors of [33] randomly swap mean and standard deviation along the spatial dimension in certain network layers, while [40] add learnable encoders and noise variables for each statistic to align them over different layers. In [30] an additional adversarial content randomization block tricks the network into predicting the right style despite changed content. Jeon *et al*. [20] sample new style vectors by defining Gaussians over the channel dimension.

**Evaluation frameworks.** Comparing algorithms based on some relevant quantitative performance metric has become the norm in the deep learning community. A common strategy for domain generalization, which by definition requires two or more different datasets, is to evaluate each algorithm on the held-out testing portion of every dataset. From the outcomes it is typically concluded that methods which, on average, perform better is superior. To account

[1] In this paper we prefer the word "texture", but interchangeably use "style" as part of already set names and expressions, as in "style transfer".

(a) ImageNet (b) Cue-Conflict (c) Edge (d) Silhouette

(e) Sketch (f) Stylized ImageNet (g) ImageNet-A (h) DeepAug EDSR and CAE

Figure 1. Samples of datasets used in our experiments. These datasets (a-f) aim to capture different facets of what is informally known as "texture bias". They attempt to decouple large scale features, such as image structure and object shapes, from small scale texture features. While humans can easily recognize the objects in these images in most cases, convolutional neural networks often struggle to do so with the same accuracy that they achieve with natural images; samples from (g) and (h) can be used to test for adversarial robustness.

for variance in learning procedures [4], a test bed similar to ours [14] trains multiple models per algorithm and uses a mean of means $\mu$ over different datasets as the final metric, i.e., method A is considered better than method B if $\mu_A > \mu_B$. Clearly, this conclusion is not necessarily justified from a statistical viewpoint [8], as it does not distinguish between true impact and random effects [4].

In fact, statistical testing offers a formal theory for comparing multiple algorithms over multiple datasets [32], a tool that is routinely used in scientific fields like physics or medicine. Hypothesis tests are employed to answer the question whether two (or more) algorithms differ significantly, given performance estimates on one or more datasets. Depending on assumptions about the scores, the comparison may require a parametric or a non-parametric test. In machine learning (ML) the requirements for parametric tests like the repeated-measures ANOVA [9] are typically not met, *e.g.*, scores may not be normally and spherically distributed [5]. Non-parametric tests like the Friedman test [10] require fewer assumptions and are more generally applicable. A recent line of work [39] proposed a Bayesian form of the Bradley-Terry model to determine how confident one can be that one algorithm is better than another.

## 3. Biased learning

Shape-biased learning tries to minimize the error that stems from distribution shifts between the training and test domains. Formally, we train a neural network $g$ on a training dataset $\{(x_i, y_i)\}_{i\in[n]}$, where the value $x_i \in \mathcal{X}$ of a random variable $X$ is the input to the network and $y_i \in \mathcal{Y}$ of a random variable $Y$ is the corresponding ground truth. Let $P(X)$ be the source domain distribution. In shape-biased learning however, we test on samples $x \in \mathcal{X}'$ such that $P(X) \neq P(X')$. In our setting the distribution shift mainly results from changes in texture, *e.g.* going from photos to cartoons or sketches. Shape-biased learning is a special case of domain generalization [26], which also encompasses distribution shifts due to factors other than texture.

Existing literature can be grouped into two families, input-augmented [13,15,27] and feature-augmented [20,22, 30, 33, 40] methods. The former create style-randomized versions of the training data, whereas the latter apply stylization to the latent representations inside a neural network. The methodology to transfer style remains the same for both. Given two images $I, I' \in \mathbb{R}^{H\times W\times 3}$ and a pretrained network (*e.g.*, VGG-19 [37]), feature statistics at suitable network layers are computed for both images. The most common approach is to use first and second moments as in [18]. Let $F_l(x) \in \mathbb{R}^{H\times W\times C_l}$ be the $l$-th feature map with input $x$, $\mu_{F_l(x)} \in \mathbb{R}^{C_l}$ its channel-wise mean (aver-

aged across the spatial dimensions) and $\sigma_{F_l(x)} \in \mathbb{R}^{C_l}$ the corresponding standard deviation. Then style transfer boils down to [18]

$$F_l^{\text{new}}(x) = \sigma_{F_l(x')}\left(\frac{x - \mu_{F_l(x)}}{\sigma_{F_l(x)}}\right) + \mu_{F_l(x')}, \qquad (1)$$

where $x, x'$ are two different samples. In case of input augmentation, the (stylized) encodings are propagated through a decoder to generate images, either offline [13] or on the fly [27]. This process is often applied to features at various representation levels to capture textures at different scales. Feature augmentation methods swap the statistics within their classification network and directly output the corresponding class predictions.

# 4. Evaluation practices

We first give an overview of current evaluation practices and point out limitations when moving out of domain. Then follows an introduction to formal hypothesis testing.

## 4.1. Current practice

Existing literature about biased learning, and also about domain generalization in a broader sense, shares common evaluation practices. Algorithms are scored on different OOD datasets in terms of an evaluation metric, usually classification accuracy, and simply averaged across datasets. From a statistical perspective, this seemingly obvious practice has several deficiencies.

First, and most importantly, often only a single point estimate of the metric is reported. However, deep learning algorithms are trained in highly stochastic fashion. Performance varies between independently trained models, since they almost certainly correspond to different local minima of the loss function. This effect is often particularly strong in the presence of domain shifts (*cf*. Fig. 2). It is evident that best practice would be to, at the very least, train multiple models with the same data and report their mean accuracy and its standard deviation. In the context of texture bias this becomes all the more important, as the variability between (and also within) training runs tends to be high.

Second, neural network training is an iterative process without a natural, unambiguous stopping criterion. Which iteration to regard as the "final" model to be evaluated is therefore invariably based on some model selection rule. Common strategies are to pick the one with the best in-domain validation score, to use a fixed iteration count, or to declare convergence based on some early-stopping rule [1, 42, 43]. Looking at the texture and shape literature, we find a lack of information about the chosen strategy. This makes a fair comparison all but impossible: texture bias schemes tend to exhibit unusually high fluctuations between training epochs, such that a different model selection rule may reverse the relative performance of two methods. This concerns all tested methods and various datasets (see Fig. 2 and Tab. 1). In this regard, we emphasize the efforts of [14], who defined a set of strategies to account for performance fluctuations and encourage multiple training runs to collect the necessary statistics for fair comparisons over numerous datasets. A diverse spectrum of image domains helps to mitigate over-fitting towards particular domains and favours generic mechanisms over dataset-specific heuristics. To obtain statistically sound conclusions it is, however, not enough to simply average scores over different datasets, as done in [14]. Naturally, different datasets possess different characteristics and as such are potentially not commensurable, so that simple averaging becomes meaningless [41]. We instead propose a rigorous, statistically founded hypothesis testing framework to compensate for the differences.

## 4.2. Hypothesis testing

This section serves as background for our experiments. We emphasize that it summarizes well-established textbook knowledge, *e.g*. [25]. The general aim is to compare $n$ algorithms $a_1, a_2, \ldots, a_n$ on $m$ datasets $d_1, d_2, \ldots, d_m$. We are interested in comparing the algorithms based on an evaluation metric $\mathcal{M}$, where $c_{ij} = \mathcal{M}(a_i, d_j)$ is the (averaged) score of algorithm $a_i$ on dataset $d_j$ (*e.g*., accuracy). Based on $c_{ij}$, we want to decide if the algorithms are statistically different [5]. We assume that the underlying metric can be expressed additively by an effect due to the algorithm and an effect due to the dataset, and $c_{ij}$ is an estimate of this underlying metric. We define the null hypothesis and alternative hypothesis as following:

$$\begin{aligned} H_0 &: \forall ik\ \gamma(a_i) = \gamma(a_k) \\ H_1 &: \exists ik\ \gamma(a_i) \neq \gamma(a_k), \end{aligned} \qquad (2)$$

where $\gamma(a_i)$ is the effect of algorithm $a_i$. In case of $n = 2$ and $m = 1$, we have a two-sample setting and with a little abuse of notation, we can write $d = \mathcal{M}(a_1, d) - \mathcal{M}(a_2, d)$ to test $H_0 : \gamma(a_1) = \gamma(a_2)$ versus $H_1 : \gamma(a_1) \neq \gamma(a_2)$.

The decision whether to reject the null hypothesis is based on the $p$-value, which is defined as the probability of obtaining a more extreme result than the one observed, assuming that $H_0$ is true:

$$p \coloneqq P(D > d \mid H_0) + P(D < d \mid H_0), \qquad (3)$$

where $D$ is the test statistic associated to our observed difference $d$. Beforehand, we must choose a (user-defined) significance level $\alpha$ and reject $H_0$ if $p < \alpha$. The setup gives rise to two (inevitable) error types: *Type I error* when rejecting $H_0$ although it is true, and *Type II error* when not rejecting $H_0$ although it is false. Here, the Type I error probability is equal to $\alpha$.

Generally, comparing multiple algorithms on multiple datasets follows a two-step procedure. First, an omnibus

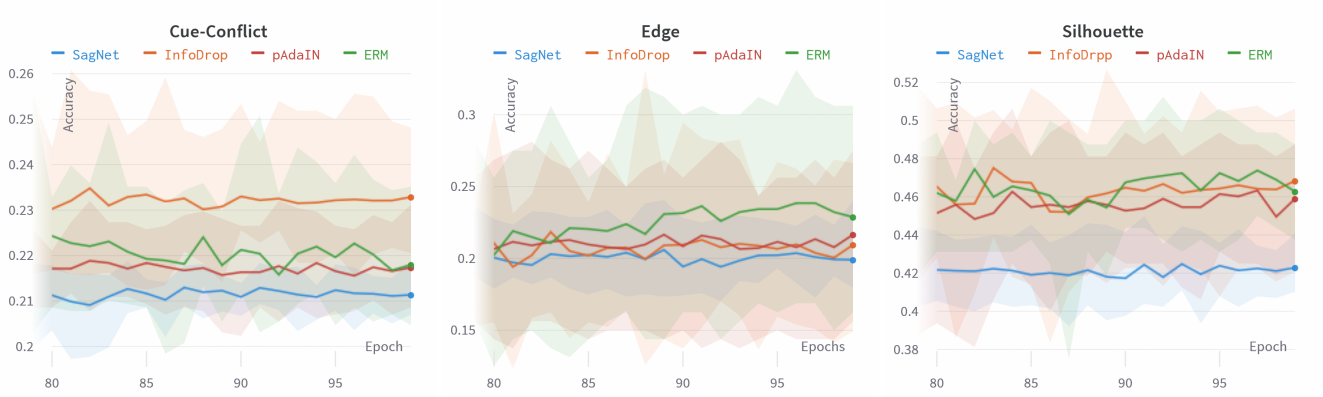


Figure 2. Accuracies of four different methods on three datasets. For each method, ten independent instances are trained and evaluated after every epoch. Lines denote average performance across then ten runs, shaded areas denote their standard deviation, highlighting significant performance fluctuations that should be taken into account.

test is applied to the hypothesis of all algorithms performing equally well, *i.e.* $H_0$ cannot be rejected. If $H_0$ can be rejected, a follow-up post-hoc test inspects each pair of algorithms to pinpoint where differences exist. A recommended omnibus test [5] in the context of ML is the Friedman test [10]. It ranks all algorithms separately in each domain (*e.g.* dataset), and uses the *average rank* $R_i$ to calculate the Friedman statistic:

$$\chi_F^2 = \frac{12m}{n(n+1)}\left[\sum_{i=1}^{n} R_i^2 - \frac{n(n+1)^2}{4}\right], \tag{4}$$

which follows approximately a $\chi^2$ distribution with $(n-1)$ degrees of freedom. The Iman-Davenport extension to the Friedman statistic compensates for too conservative decisions and is defined as

$$F_F = \frac{(m-1)\chi_F^2}{m(n-1)-\chi_F^2}, \tag{5}$$

which is approximately distributed according to an $F$-distribution with $n-1$ and $(m-1)(n-1)$ degrees of freedom. If the $p$-value is below the pre-defined significance threshold $\alpha$, the Nemenyi post-hoc test [31] compares pairs of differences. The Nemenyi test statistic for algorithms $a_i$ and $a_j$ is defined as

$$z = |R_i - R_j| / \sqrt{\frac{m(m+1)}{6n}}. \tag{6}$$

For large values of $z$ the difference is significant. Note that the two-step procedure is necessary to maintain the overall Type I error probability, which is not the case when testing all pairs within a two-sample framework at level $\alpha$.

## 5. BiasBed framework

We introduce *BiasBed*, a highly flexible and generic test suite to implement, train and evaluate existing algorithms in a rigorous manner. We currently include seven texture debiasing algorithms, ten datasets, different model selection strategies and an omnibus hypothesis test and a post-hoc hypothesis test. Adding existing and new algorithms, other datasets or running a hyperparameter search for one algorithm is a matter of a few lines. In fact, our framework provides a full integration with Weights and Biases [3] providing the full range of logging, visualization and reports.

**Algorithms.** *BiasBed* currently supports several algorithms that are tailored towards shape biased learning. Our baseline algorithm is a plain ResNet-50 trained with standard empirical risk minimization [38]. The same algorithm can be trained on Stylized ImageNet to duplicate the approach by Geirhos *et al.* [13]. Additional algorithms are: Shape-Texture Debiased Neural Network (Debiased, [27]), Style-Agnostic Network (SagNet, [30]), Permuted Adaptive Instance Normalization (pAdaIN, [33]), Informative Dropout (InfoDrop, [36]) and Deep Augmentation (DeepAug, [15]). We show in the Appendix how to easily add more algorithms besides the ones listed here.

**Datasets.** Besides the standard dataloader for ImageNet1k [6], *BiasBed* contains dataloaders for five additional texture-bias datasets and four robustness datasets. Stylized ImageNet [13] and Cue-Conflict [12] are stylized versions of ImageNet (1000 classes) and 16-class ImageNet, respectively, where the latter particularly uses texture cues for stylization. Datasets that contain pure shape information are Edges, Sketches and Silhouettes from [12]. We further extend BiasBed by including the following robustness datasets: ImageNet-A [16], ImageNet-R [15] and two versions of DeepAug [15].

**Model selection.** In Section 4, we have seen that when testing on out-of-distribution datasets, where performance fluctuations during training are rather high, the criterion used to pick the final model can have severe impacts on the final quality metrics. Therefore, we have implemented three model selection methods in *BiasBed* to unify evaluation.

- *Best-epoch* is an oracle method that chooses the best test set score over all epochs after training for a fixed number of epochs. Importantly, this method is *cherry-picking* and should be discouraged to use. We include it here to highlight the severe effects of model selection on evaluation.
- *Last-n-epochs* averages the test set scores over the last $n$ epochs, to mitigate the fluctuations observed in the OOD regime. *I.e.*, once the training has converged the model is evaluated after every training epoch, and a representative average score is reported.
- *Best-training-validation* chooses a single test set score according that achieves the best *in-domain* performance on the validation set. This method is the most rigorous one, but leads to the highest variance between independent test runs. To decrease that variance, more runs are needed.

**Evaluation.** *BiasBed* includes a full, rigorous evaluation protocol that can be run with a single command. It collects the result according to the defined model selection method, algorithms and datasets, and computes the averaged scores. In a second step, the Friedman test with Iman-Davenport extension [19] is applied with a possible post-hoc Nemenyi test to identify significant differences. All results are reported back in a Pandas dataframe [29, 34] or optionally in LaTeX tables (as those in Section 6).

# 6. Experiments

We run experiments for all implemented algorithms and extensively compare different model selection methods, highlighting the need for a common, explicit protocol. We use hypothesis testing to properly compare across algorithms. The following results are all generated by running a single command for each algorithm in *BiasBed*.

## 6.1. Model selection

We quantitatively elaborate on the findings of Fig. 2. We train each algorithm ten times and test on *Silhouette*, *Edge* and *Cue-Conflict* after *every* epoch during training. In Table 2 we report the best accuracy over all epochs averaged across runs versus the last score received, averaged across all runs. Clearly, all algorithms drop severely in performance when one does not use an oracle model selection method, highlighting the need to properly define criteria how to select the model to be used. We emphasize that this step should always be part of an evaluation.

| Algorithm | Silhouette | Edge | Cue-Conflict |
|---|---|---|---|
| ERM | 50.8 / 47.3 (-3.5) | 32.6 / 22.6 (-10.0) | 26.4 / 22.2 (-4.2) |
| SagNet | 48.1 / 43.3 (-4.8) | 32.7 / 25.3 (-7.4) | 21.3 / 20.0 (-1.3) |
| pAdaIN | 48.0 / 43.8 (-4.2) | 29.0 / 22.3 (-6.7) | 24.0 / 21.5 (-2.5) |
| InfoDrop | 51.2 / 47.8 (-3.4) | 31.7 / 19.0 (-11.7) | 26.6 / 22.8 (-3.8) |

Table 1. Comparison of best epoch (oracle) *vs*. checkpoint with best validation accuracy for selected algorithms and datasets.

## 6.2. Results

**Evaluation.** We report results choosing the best validation performance for all algorithms and datasets in Table 2 and choosing the last 30 epochs in Table 3. All results are once more gathered across multiple independent runs to account for randomness in training. In particular, we do not include an average column across dataset scores per algorithm, as discussed in Section 4. Note that in three cases data used for training was augmented in a similar way as the dataset used for testing, *i.e.* there is no domain shift between train and test distribution. Naturally, performance is significantly higher. These results are underlined to emphasize this fact.

**Hypothesis testing.** We conduct the Friedman test on the scores of Table 2. We conduct hypothesis tests in two ways: once with all datasets, and another more strict setting where we reject datasets which were created using the same methods as for training any of the algorithms. We chose a significance level of $\alpha = 0.05$ and test for significance. The returned F-statistic is $7.46$ with an uncorrected $p$-value of $0.000002$ when using all datasets and $F = 4.98$, $p = 0.00045$ in the strict setting. In both cases, $p < \alpha$ and therefore we reject $H_0$. In Table 4 and Table 5 we report all pairwise comparisons of algorithms using the Nemenyi post-hoc test for both settings.

# 7. Discussion

Our results, found in Tabs. 1, 2, 3, and most importantly Tab. 4 and Tab. 5, allow us to draw several conclusions that support the usage of the formal hypothesis testing framework for this evaluation.

**Input augmentation and feature augmentation.** The considered algorithms can be grouped into two main families: those that focus on augmenting the inputs that go into the network, and those that focus on the activations inside the network. We observe in Tab. 2 and Tab. 3 that algorithms that perform input augmentation (Stylized ERM, Debiased and both DeepAug ERMs) tend to have larger differences in

Table 2. Results choosing the best validation accuracy per run. Columns are grouped according to texture bias and adversarial robustness. Datasets marked with † are those where an algorithm has been trained on the corresponding dataset, *i.e*. no distribution gap in the test set. The specific score is underlined. Input augmented and feature augmented methods are grouped together.

| Algorithm | ImageNet1k | Silhouette | Edge | Sketch | Cue-Conflict | Stylized ImageNet† | ImageNet-A | ImageNet-R | DeepAug (CAE)† | DeepAug (EDSR)† |
|---|---|---|---|---|---|---|---|---|---|---|
| ERM [38] | 73.8 ± 0.2 | 47.3 ± 2.4 | 22.6 ± 3.3 | 56.0 ± 1.0 | 22.2 ± 0.9 | 7.9 ± 0.2 | 2.0 ± 0.2 | 22.8 ± 0.4 | 44.4 ± 0.3 | 48.3 ± 0.5 |
| pAdaIN [33] | 73.2 ± 0.1 | 43.8 ± 2.2 | 22.3 ± 1.7 | 56.6 ± 0.8 | 21.5 ± 0.6 | 8.1 ± 0.1 | 1.4 ± 0.1 | 21.4 ± 0.3 | 42.8 ± 0.2 | 48.8 ± 0.3 |
| SagNet [30] | 74.2 ± 0.5 | 43.3 ± 1.6 | 25.3 ± 2.0 | 59.2 ± 1.1 | 20.0 ± 0.2 | 6.2 ± 0.1 | 1.5 ± 0.2 | 21.8 ± 0.5 | 43.8 ± 0.4 | 47.7 ± 0.5 |
| InfoDrop [36] | 73.3 ± 0.2 | 47.8 ± 2.8 | 19.0 ± 4.3 | 56.7 ± 2.0 | 22.8 ± 0.6 | 7.9 ± 0.3 | 2.2 ± 0.1 | 22.7 ± 0.3 | 44.4 ± 0.2 | 48.6 ± 0.5 |
| Stylized ERM [13] | 55.9 ± 0.5 | 46.9 ± 2.8 | 58.4 ± 3.1 | 70.2 ± 1.4 | 53.7 ± 1.4 | 53.2 ± 0.2 | 0.8 ± 0.1 | 25.0 ± 0.3 | 39.1 ± 0.5 | 40.4 ± 0.4 |
| Debiased [27] | 74.4 ± 0.1 | 48.7 ± 2.8 | 30.8 ± 5.1 | 60.5 ± 1.2 | 28.9 ± 1.1 | 16.1 ± 0.3 | 2.7 ± 0.2 | 27.4 ± 0.4 | 49.6 ± 0.2 | 51.3 ± 0.3 |
| DAug. ERM (CAE) [15] | 73.7 ± 0.2 | 51.3 ± 3.2 | 34.7 ± 7.4 | 63.5 ± 2.6 | 29.9 ± 2.3 | 12.7 ± 2.0 | 2.6 ± 0.2 | 27.8 ± 1.7 | 61.4 ± 5.8 | 55.8 ± 2.9 |
| DAug. ERM (EDSR) [15] | 72.8 ± 0.2 | 51.6 ± 1.3 | 31.7 ± 4.6 | 61.1 ± 1.5 | 32.4 ± 1.5 | 11.0 ± 0.3 | 2.0 ± 0.2 | 26.5 ± 0.5 | 52.3 ± 0.3 | 65.1 ± 0.2 |

Table 3. Results choosing the last 30 epochs per run. Columns are grouped according to texture bias and adversarial robustness. Datasets marked with † are those where an algorithm has been trained on the corresponding dataset, *i.e*. no distribution gap in the test set. The specific score is underlined. Input augmented and feature augmented methods are grouped together.

| Algorithm | ImageNet1k | Silhouette | Edge | Sketch | CueConflict | Stylized ImageNet† | ImageNet-A | ImageNet-R | DeepAug (CAE)† | DeepAug (EDSR)† |
|---|---|---|---|---|---|---|---|---|---|---|
| ERM [38] | 73.3 ± 0.4 | 46.8 ± 2.4 | 22.6 ± 3.7 | 56.3 ± 1.3 | 22.1 ± 0.9 | 7.7 ± 0.3 | 2.1 ± 0.2 | 22.5 ± 0.5 | 43.8 ± 0.5 | 47.8 ± 0.6 |
| pAdaIN [33] | 73.1 ± 0.1 | 44.5 ± 2.1 | 22.1 ± 2.9 | 56.5 ± 0.9 | 21.4 ± 0.7 | 8.1 ± 0.2 | 1.5 ± 0.1 | 21.4 ± 0.3 | 42.7 ± 0.3 | 48.6 ± 0.3 |
| SagNet [30] | 73.9 ± 0.5 | 44.6 ± 1.4 | 27.1 ± 2.4 | 60.4 ± 1.2 | 19.7 ± 0.5 | 6.2 ± 0.2 | 1.7 ± 0.2 | 22.5 ± 0.5 | 43.2 ± 0.7 | 47.3 ± 0.5 |
| InfoDrop [36] | 72.9 ± 0.5 | 47.0 ± 2.7 | 18.9 ± 4.4 | 56.6 ± 1.7 | 22.9 ± 0.8 | 7.7 ± 0.3 | 2.2 ± 0.2 | 22.7 ± 0.5 | 43.9 ± 0.6 | 48.4 ± 0.6 |
| Stylized ERM [13] | 55.3 ± 0.7 | 47.3 ± 2.6 | 59.5 ± 3.5 | 70.2 ± 1.2 | 54.2 ± 1.5 | 52.4 ± 0.6 | 0.7 ± 0.1 | 25.0 ± 0.5 | 38.8 ± 0.7 | 40.1 ± 0.6 |
| Debiased [27] | 74.0 ± 0.3 | 48.3 ± 2.8 | 29.4 ± 4.9 | 60.1 ± 1.3 | 29.1 ± 1.1 | 15.6 ± 0.5 | 2.6 ± 0.2 | 27.2 ± 0.5 | 49.1 ± 0.5 | 50.8 ± 0.6 |
| DAug. ERM (CAE) [15] | 72.8 ± 0.3 | 50.5 ± 3.4 | 33.1 ± 7.3 | 62.6 ± 2.9 | 29.4 ± 2.4 | 12.4 ± 1.9 | 2.7 ± 0.2 | 27.0 ± 1.7 | 60.6 ± 5.9 | 55.2 ± 2.9 |
| DAug. ERM (EDSR) [15] | 71.9 ± 0.7 | 51.4 ± 2.2 | 32.0 ± 4.2 | 60.4 ± 1.9 | 32.3 ± 1.2 | 10.6 ± 0.5 | 2.0 ± 0.2 | 25.8 ± 0.6 | 51.1 ± 0.9 | 64.2 ± 0.8 |

performance w.r.t. ERM (trained solely on ImageNet) than algorithms that try to mitigate style bias through latent space (feature) augmentation (SagNet, InfoDrop, pAdaIN). This suggests that the proposed feature augmentations – such as using AdaIN to shift and scale feature maps – fail to capture real variations in texture. We hypothesize that such augmentations require the decoder from a pre-trained autoencoder to better express texture features. However, and importantly, only in rare cases do the algorithms statistically differ from each other (*cf*. Tab. 4, Tab. 5). In fact, taking into account only the true OOD datasets none of the implemented algorithms significantly outperforms a baseline ERM.

**Model selection criteria.** If model selection is done based on validation accuracy, the margins are extremely small compared to the uncertainty, as measured by the standard deviation over 10 different runs of an algorithm. This suggests that, while validation performance is of course to some degree predictive of OOD performance, it apparently falls short of capturing all relevant effects that impact such generalization, and is not sufficient. It seems clear that having an explicit, formal model selection strategy is of paramount importance. We can also conclude that reporting results for a single run is not a reliable way of comparing different approaches, as the stochastic nature of the training alone can flip the relative performance of two methods.

**Intra-run and inter-run variability.** While some authors acknowledge variability in the results across different runs, it is rare to find statements about intra-run variability, *i.e*., strong performance fluctuations between nearby training checkpoints, as is depicted by the shaded regions in Fig. 2. It is sometimes the case that authors report the mean and standard deviation of performance metrics for several runs to mitigate this, but the significant variations not only between independent training runs, but also among different epochs of an apparently converged training mean that averages without the associated uncertainty are problematic, and it is nearly impossible to draw reliable conclusions without a formal framework. The complex interplay between these factors needs to be acknowledged and analysed more closely. These types of variability stem from different sources and need to be handled in different ways.

**Importance of hypothesis testing.** The presented results also highlight the importance of using a formal comparison framework when dealing with such complex cases. When informally analysing Tables 2 and 3, such as by simply averaging each algorithm's performance across datasets, it is easy to arrive at erroneous conclusions based on spurious results. For instance, the results seen in Tab. 2 could lead one to believe that the Debiased and DeepAug algorithms do outperform competitors due to their high performance on datasets such as Sketch, CueConflict, and Sylized IN, with little or no performance loss on ImageNet1k when compared to ERM, the baseline result. But this analysis does not take into account the variances of these results and the complex interplay between the different measured accuracies, sample sizes, *etc*. In fact, the results of the post-hoc Nemenyi test, reported in Tab. 4 and Tab. 5, tell us that these algorithms do not differ from ERM in a statistically significant way – one can not refute that null hypothesis that

Table 4. Full post-hoc Nemenyi test based on validation. For this hypothesis test we use scores from all datasets.

| Algorithm | ERM | pAdaIN | SagNet | InfoDrop | Stylized ERM | Debiased | DAug. ERM (CAE) | DAug. ERM (EDSR) |
|---|---|---|---|---|---|---|---|---|
| ERM [38] | 1.0 | 0.9 | 0.9 | 0.9 | 0.9 | 0.211 | 0.053 | 0.478 |
| pAdaIN [33] | | 1.0 | 0.9 | 0.9 | 0.642 | 0.012 | 0.002 | 0.053 |
| SagNet [30] | | | 1.0 | 0.9 | 0.806 | 0.03 | 0.005 | 0.111 |
| InfoDrop [36] | | | | 1.0 | 0.9 | 0.254 | 0.069 | 0.533 |
| Stylized ERM [13] | | | | | 1.0 | 0.642 | 0.303 | 0.9 |
| Debiased [27] | | | | | | 1.0 | 0.9 | 0.9 |
| DAug. ERM (CAE) [15] | | | | | | | 1.0 | 0.9 |
| DAug. ERM (EDSR) [15] | | | | | | | | 1.0 |

Table 5. Partial post-hoc Nemenyi test based on validation. This hypothesis test is based solely on datasets that have a true distribution shift wrt. the training data.

| Algorithm | ERM | pAdaIN | SagNet | InfoDrop | Stylized ERM | Debiased | DAug. ERM (CAE) | DAug. ERM (EDSR) |
|---|---|---|---|---|---|---|---|---|
| ERM [38] | 1.0 | 0.897 | 0.9 | 0.9 | 0.9 | 0.505 | 0.241 | 0.897 |
| pAdaIN [33] | | 1.0 | 0.9 | 0.897 | 0.364 | 0.024 | 0.005 | 0.149 |
| SagNet [30] | | | 1.0 | 0.9 | 0.897 | 0.241 | 0.086 | 0.636 |
| InfoDrop [36] | | | | 1.0 | 0.9 | 0.505 | 0.241 | 0.897 |
| Stylized ERM [13] | | | | | 1.0 | 0.9 | 0.766 | 0.9 |
| Debiased [27] | | | | | | 1.0 | 0.9 | 0.9 |
| DAug. ERM (CAE) [15] | | | | | | | 1.0 | 0.9 |
| DAug. ERM (EDSR) [15] | | | | | | | | 1.0 |

the two methods are on par. It is possible that using more datasets would allow us to identify for which algorithms, if any, there is astatistically significant difference, but even with the rather many runs in our test bed, no statistically supported difference has been observed yet, and we cannot confidently establish a correct ranking.

**Interpretation of hypothesis testing.** When using hypothesis tests, one needs to be precise w.r.t. the conclusions drawn from the results. In particular, rejecting the null hypothesis means that if the null hypothesis were correct, we would almost certainly (with significance $\alpha$) not observe such an extreme difference. However, even if the significance level is high we cannot conclude superiority of one algorithm over another for an unseen domain. The *significance* of the difference is only established under the experimental setting, *i.e.* within the datasets and algorithms used. Conversely, being unable to reject the null hypothesis should not be misinterpreted as a proof of equality, but only as an inability to rule out equality with the given data.

## 8. Conclusion

When analyzing existing methods tailored towards texture-free learning, common datasets, evaluation protocols and reports are missing. In this work, we introduced *BiasBed* to alleviate the aforementioned limitations. In particular, we have seen that model selection methods play a critical role in OOD testing and fair comparisons are only possible if algorithms are evaluated in a rigorous fashion. Hypothesis testing can fill this gap by providing statistically sound comparisons. Moreover, one must be careful to draw the right conclusions, *e.g.*, low significance of a difference does not necessarily mean that two methods perform on par, but may also indicate that there are too few observations to make a confident statement about their difference. Our framework provides the necessary tools to implement, test and compare existing and new algorithms. Our intention is not to make negative claims or invalidate any particular approach. Rather, we hope to encourage the community to leverage existing statistical expertise and ensure fair and rigorous quantitative evaluations that drive forward the field.

## 9. Broader impact

The aim of the work presented in this paper is to provide a solid foundation for future research on style bias of neural networks. Such hypothesis testing framework are commonplace in other fields where the effects of different experiments can only be observed in noisy results, such as many areas of physics, medicine and psychology. We hope that the presented framework will be used in the future by other researchers through the openly released codebase to find and validate novel algorithms that mitigate texture bias. Furthermore, the proposed methodology – and codebase – can be used to perform rigorous testing and comparisons of algorithms for other computer vision and machine learning problems where results are notoriously hard to validate, such as domain adaptation and domain generalization. It is our belief that setting a higher bar and expecting rigorous testing from authors who propose novel methods will, in the long run, improve the quality of research and advance the field.

## References


[1] Bowen Baker, Otkrist Gupta, Ramesh Raskar, and Nikhil Naik. Accelerating neural architecture search using performance prediction. *arXiv preprint arXiv:1705.10823*, 2017. 4

[2] Moshe Bar, Karim S Kassam, Avniel Singh Ghuman, Jasmine Boshyan, Annette M Schmid, Anders M Dale, Matti S Hämäläinen, Ksenija Marinkovic, Daniel L Schacter, Bruce R Rosen, et al. Top-down facilitation of visual recognition. *Proceedings of the National Academy of Sciences*, 103(2):449–454, 2006. 1

[3] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com. 5

[4] Xavier Bouthillier, Pierre Delaunay, Mirko Bronzi, Assya Trofimov, Brennan Nichyporuk, Justin Szeto, Nazanin Mohammadi Sepahvand, Edward Raff, Kanika Madan, Vikram Voleti, et al. Accounting for variance in machine learning benchmarks. *Proceedings of Machine Learning and Systems*, 3:747–769, 2021. 3

[5] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006. 3, 4, 5

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 5

[7] James J DiCarlo, Davide Zoccolan, and Nicole C Rust. How does the brain solve visual object recognition? *Neuron*, 73(3):415–434, 2012. 1

[8] Rotem Dror, Gili Baumer, Marina Bogomolov, and Roi Reichart. Replicability analysis for natural language processing: Testing significance with multiple datasets. *Transactions of the Association for Computational Linguistics*, 5:471–486, 2017. 3

[9] Ronald Aylmer Fisher. Statistical methods for research workers. In *Breakthroughs in Statistics*, pages 66–70. Springer, 1992. 3

[10] Milton Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937. 3, 5

[11] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Texture and art with deep neural networks. *Current Opinion in Neurobiology*, 46:178–186, 2017. 1, 2

[12] Robert Geirhos, Kantharaju Narayanappa, Benjamin Mitzkus, Tizian Thieringer, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Partial success in closing the gap between human and machine vision. *NeurIPS*, 34, 2021. 1, 5

[13] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*, 2018. 1, 2, 3, 4, 5, 7, 8

[14] Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. *arXiv preprint arXiv:2007.01434*, 2020. 3, 4

[15] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. *ICCV*, 2021. 1, 3, 5, 7, 8

[16] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. *CVPR*, 2021. 5

[17] Katherine Hermann, Ting Chen, and Simon Kornblith. The origins and prevalence of texture bias in convolutional neural networks. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *NeurIPS*, 2020. 1

[18] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017. 3, 4

[19] Ronald L Iman and James M Davenport. Approximations of the critical region of the Friedman statistic. *Communications in Statistics-Theory and Methods*, 9(6):571–595, 1980. 6

[20] Seogkyu Jeon, Kibeom Hong, Pilhyeon Lee, Jewook Lee, and Hyeran Byun. Feature stylization and domain-aware contrastive learning for domain generalization. In *ACM International Conference on Multimedia*, pages 22–31, 2021. 1, 2, 3

[21] Nikolai Kalischek, Jan D Wegner, and Konrad Schindler. In the light of feature distributions: moment matching for neural style transfer. In *CVPR*, 2021. 2

[22] Juwon Kang, Sohyun Lee, Namyup Kim, and Suha Kwak. Style neophile: Constantly seeking novel styles for domain generalization. In *CVPR*, 2022. 1, 2, 3

[23] Nikolaus Kriegeskorte. Deep neural networks: a new framework for modeling biological vision and brain information processing. *Annual Review of Vision Science*, 1:417–446, 2015. 2

[24] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. 2

[25] Erich Leo Lehmann. *Testing statistical hypotheses*, volume 2. Springer, 1986. 4

[26] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *ICCV*, 2017. 3

[27] Yingwei Li, Qihang Yu, Mingxing Tan, Jieru Mei, Peng Tang, Wei Shen, Alan Yuille, and Cihang Xie. Shape-texture debiased neural network training. *arXiv preprint arXiv:2010.05981*, 2020. 1, 2, 3, 4, 5, 7, 8

[28] Nikos K Logothetis and David L Sheinberg. Visual object recognition. *Annual Review of Neuroscience*, 19(1):577–621, 1996. 1

[29] Wes McKinney. Data structures for statistical computing in python. In *Python in Science Conference*, 2010. 6

[30] Hyeonseob Nam, HyunJae Lee, Jongchan Park, Wonjun Yoon, and Donggeun Yoo. Reducing domain gap by reducing style bias. In *CVPR*, 2021. 1, 2, 3, 5, 7, 8

[31] Peter Bjorn Nemenyi. *Distribution-free multiple comparisons*. Princeton University, 1963. 5

[32] Jerzy Neyman and Egon S Pearson. On the use and interpretation of certain test criteria for purposes of statistical inference: Part i. *Biometrika*, pages 175–240, 1928. 3

[33] Oren Nuriel, Sagie Benaim, and Lior Wolf. Permuted adain: Reducing the bias towards global statistics in image classification. In *CVPR*, 2021. 1, 2, 3, 5, 7, 8

[34] The pandas development team. pandas-dev/pandas: Pandas, Feb. 2020. 6

[35] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *NeurIPS*, 32, 2019. 2

[36] Baifeng Shi, Dinghuai Zhang, Qi Dai, Zhanxing Zhu, Yadong Mu, and Jingdong Wang. Informative dropout for robust representation learning: A shape-bias perspective. In *ICML*, 2020. 5, 7, 8

[37] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 3

[38] Vladimir N Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999, 1999. 5, 7, 8

[39] Jacques Wainer. A Bayesian Bradley-Terry model to compare multiple ML algorithms on multiple data sets. *arXiv preprint arXiv:2208.04935*, 2022. 3

[40] Yue Wang, Lei Qi, Yinghuan Shi, and Yang Gao. Feature-based style randomization for domain generalization. *IEEE Transactions on Circuits and Systems for Video Technology*, 2022. 1, 2, 3

[41] Geoffrey I Webb. Multiboosting: A technique for combining boosting and wagging. *Machine Learning*, 40(2):159–196, 2000. 4

[42] Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. On early stopping in gradient descent learning. *Constructive Approximation*, 26(2):289–315, 2007. 4

[43] Tong Zhang and Bin Yu. Boosting with early stopping: Convergence and consistency. *Annals of Statistics*, 33(4):1538–1579, 2005. 4