On Measuring Localization of Shortcuts in Deep Networks

Anonymous authors
Paper under double-blind review

Abstract

Shortcuts, spurious rules that perform well during training but fail to generalize, present a major challenge to the reliability of deep networks (Geirhos et al., 2020). To design principled shortcut-mitigation methods, it is crucial to understand how shortcuts affect feature representations. In this work, we investigate the layer-wise localization of shortcuts in deep models. We propose a novel experiment design that quantifies the layer-wise contribution to accuracy degradation caused by a shortcut. Our method introduces a shortcut-inducing data skew into the training process and counterfactually compares training on clean and skewed datasets using suitable shortcut-learning metrics. We employ our method to study vision classification shortcuts across the CIFAR-10, Waterbirds, and CelebA datasets and the VGG, ResNet, DeiT, and ConvNeXt architectures. We find that shortcut learning is not localized in specific layers but distributed throughout the network. Different network parts play different roles in this process: earlier layers predominantly encode spurious features, while later layers predominantly forget core features (i.e., features that are predictive on clean data). We analyze the differences in localization and describe their principal axes of variation. Finally, we investigate layer-wise training interventions and find that our localization metrics are predictive of their success.

1 Introduction

Shortcuts, spurious rules that hold within training distributions but fail to generalize to real-world scenarios, present a major challenge to the reliability of deep models. Despite their significance, the mechanisms of shortcut learning remain poorly understood. While shortcuts arise from the statistical phenomenon of spurious correlations (Arjovsky et al., 2020), it remains unclear how and which correlations are captured by networks (Hermann & Lampinen, 2020).

A crucial yet underexplored dimension of shortcut learning is the hierarchical nature of deep networks. Since different layers correspond to distinct levels of abstraction and feature complexity (Simonyan et al., 2014), shortcuts likely manifest differently across layers. Quantifying the impact of this phenomenon on accuracy could inform the design of layer-specific shortcut mitigations (e.g., Lee et al., 2023). However, existing research falls short in this regard—either focusing solely on overall accuracy without layer-wise effects (Scimeca et al., 2022) or analyzing representations without explicitly connecting these findings to performance (Hermann & Lampinen, 2020; Islam et al., 2021).

Contributions To bridge this gap, we develop a method for quantifying the layer-wise effects of shortcuts on model accuracy. Our approach measures *each layer's contribution to shortcut learning*, expressed as accuracy degradation on clean test data, by analyzing *counterfactual changes* in network behavior when trained on skewed and clean data. We decompose shortcut learning into two fundamental processes: spurious feature promotion and core feature degradation. We analyze them using two metrics: *spurious feature encoding and core feature forgetting.*

We apply our methodology to study a watermark skew on CIFAR-10, a background skew on Waterbirds, and a sampling skew on CelebA across the VGG-11, ResNet-18, DeiT-Ti, and ConvNeXt-T architectures. Our findings reveal that shortcut learning is not localized in specific layers but is instead distributed throughout the entire network. Different layers play different roles in shortcut learning: earlier layers mostly contribute

to spurious feature encoding, while later layers mostly contribute to core feature forgetting. Dataset and model factors explain 83.8% of variance in encoding localization, while data skew frequency and optimizer factors explain 57.0% of variance in forgetting localization in fine-tuned models. Finally, we find that our metrics predict the success of some layer-wise interventions.

Overall, our localization results and factor analysis provide insights into shortcut learning in deep networks and allow us to speculate about the roles of different layers in this process. Our localization metrics predict the success of certain layer-wise interventions (e.g., layer freezing) in shortcut mitigation. However, the metrics also vary widely across the considered learning settings. Together, these findings suggest that the effects of layer-wise interventions vary widely across learning settings, highlighting the difficulty of designing layer-wise shortcut-mitigation strategies.

2 Related work

Impact of shortcuts on feature representations We expand the existing literature on the impact of shortcuts on representations. Previous studies analyze how shortcuts are encoded in layers using linear probing accuracy (Hermann & Lampinen, 2020), mutual information and read-out module accuracy (Islam et al., 2021), or validation accuracy on feature-labeled datasets (Scimeca et al., 2022). While these approaches provide valuable insights, they do not explicitly attribute the degradation in accuracy to specific layers. In contrast, our work directly quantifies *layer-wise contributions to the degradation in accuracy.*

Quantification of layers' importance Similarly to us, several works investigate feature representations in deep models and assess the importance of each layer for specific model properties. Zhang et al. (2022) measure the importance of each layer of a deep network for classification accuracy by injecting noise in the network weights. Maini et al. (2023) analyze the memorization behavior of different layers by introducing label noise. Huh et al. (2023) analyze learned feature representations of deep models and show how certain properties of these representations support generalization. While these studies provide valuable insights into feature learning, they do not specifically study shortcut learning.

Mechanisms of shortcut formation Our work contributes to the literature on shortcut learning mechanisms (Shah et al., 2020; Sagawa et al., 2020b; Nagarajan et al., 2021; Chaudhuri et al., 2023; Puli et al., 2023; Wang et al., 2023; Tsoy & Konstantinov, 2024). Our analysis suggests that feature forgetting plays a key role in shortcut learning, supporting prior hypotheses that simplicity bias (e.g., Shah et al., 2020) or excessive regularization (Sagawa et al., 2020b) are important for shortcut formation. In contrast to these works, we measure the contributions of different parts of the network to shortcut learning, contributing to a fine-grained quantitative understanding of shortcuts.

Layer-wise fine-tuning analysis Our work is related to the literature on layer-wise adaptation of deep models to distribution shifts (e.g., Kumar et al., 2022; Lee et al., 2023; Trivedi et al., 2023; Kirichenko et al., 2023). Our findings help to reason about fine-tuning strategies for shortcut mitigation (see Section 6).

3 Methodology

This section outlines our method for measuring *layer-wise contributions* to shortcut learning. Our approach introduces controlled *shortcut-inducing data skews* into the training process, such as replacing the background of an image with a class-correlated one, like in Waterbirds (Sagawa et al., 2020a). This approach allows us to assess each layer's role *counterfactually* by controlling all training factors except the data itself. We train multiple networks on the same task, exposing different blocks of layers to either skewed or clean (skew-free) data and evaluate these networks on a clean test dataset to quantify each block's contribution.

The two main components of our method are shortcut learning metrics and a counterfactual training algorithm. After briefly outlining these components below, we present the main idea behind our method and motivate our shortcut-learning metrics using an example of a two-layer architecture consisting of a feature extractor and a classifier (Section 3.1). Then, we adapt our metrics to the general case in Section 3.2. Fi-

nally, in Section 3.3, we present a counterfactual training algorithm that allows us to compute our metrics for neural networks trained with gradient methods.

Shortcut learning metrics To measure how spurious and core features are localized and used, we use two metrics: *spurious feature encoding* and *core feature forgetting*. For the two-layer architecture, they determine how much the feature extractor "incentivizes" the classifier to rely on spurious features and how much the feature extractor "disincentivizes" the classifier to rely on core features. We aim to localize the contributions of blocks to these phenomena to better understand the mechanisms of shortcut learning.

Counterfactual training algorithm A crucial aspect for objectively measuring these metrics is the design of an appropriate training scheme. Our scheme needs to maintain consistency between the learning mechanisms in the original and skewed blocks, i.e., it should *control for all factors other than the training data itself* to avoid biases in localization measures. At the same time, due to overparametrization (Brunet et al., 2022) and non-convexity, deep learning is highly sensitive to even small changes in training procedure, making this task non-trivial.

3.1 Illustrative example

Assume that we have two classification datasets: clean D^c and skewed $D^s = g(D^c)$, where g is a shortcut-inducing data skew. The labels in both datasets are determined by core features that are generalizable across both classification tasks. However, the labels in the skewed dataset are also correlated with spurious features that are not generalizable to the clean task but can be used to classify skewed data. Except for the presence of spurious features, the datasets are very similar: they have the same number of points and correspond to the same distribution of core features.

We train an architecture $h = c \circ f$ consisting of a classifier c and a feature extractor f on these datasets. This process results in two models $h^c = c^c \circ f^c$ trained on clean data and $h^s = c^s \circ f^s$ trained on skewed data. Since the skewed dataset can be classified using both clean and spurious features, we expect the skewed model to learn shortcut rules that rely on spurious features, leading to brittle generalization.

We use an increase in test error rate on *clean data*, $\mathrm{err}(h^s) - \mathrm{err}(h^c)$, as a measure for *shortcut reliance*. Note that $\mathrm{err}(h^c)$ accounts for approximation error (i.e., architectural limitations), optimization error, and finite sample error. At the same time, $\mathrm{err}(h^s)$ accounts for all these factors and, in addition, includes the error due to shortcut reliance since the labels on clean data do not depend on spurious features. Since we will compare the same architectures, trained with the same optimization procedure, on the datasets with the same number of points, the considered difference corresponds exactly to the error due to shortcut reliance.

We aim to quantify the contributions of the classifier c and feature extractor f to the increase in test error rate $\mathrm{err}(h^s) - \mathrm{err}(h^c)$. To achieve this goal, we consider two classifiers: ${c^c}_{f^s}$, a classifier adapted to classify clean data using skewed extractor f^s, and ${c^s}_{f^c}$, a classifier adapted to classify skewed data using clean extractor f^c. We get the following decompositions:

$$\begin{aligned}
&\mathrm{err}(h^s) - \mathrm{err}(h^c) \\
&= \mathrm{err}(c^s \circ f^s) - \mathrm{err}({c^c}_{f^s} \circ f^s) + \mathrm{err}({c^c}_{f^s} \circ f^s) - \mathrm{err}(c^c \circ f^c) \\
&= \mathrm{err}(c^s \circ f^s) - \mathrm{err}({c^s}_{f^c} \circ f^c) + \mathrm{err}({c^s}_{f^c} \circ f^c) - \mathrm{err}(c^c \circ f^c).
\end{aligned} \tag{1}$$

Shortcut learning metrics Consider the term $\mathrm{err}({c^c}_{f^s} \circ f^s) - \mathrm{err}(c^c \circ f^c)$. Here, both classifiers are trained on the same clean data, while the extractors differ. Therefore, this difference isolates the effect of replacing the clean extractor with the skewed one. Since the clean data contain no systematic statistical relationship between spurious features and labels, the classifiers are incentivized to rely on core features. If the classifier adapted to the skewed extractor performs worse, this indicates that the skewed extractor makes core features less prominent to the classifier (e.g., by shrinking core-feature amplitudes or losing core information). For this reason, we say that this term measures *core feature forgetting* by the skewed feature extractor relative to the clean one.

Similarly, consider the term $\mathrm{err}(c^s \circ f^s) - \mathrm{err}(c^s{}_{f^c} \circ f^c)$. Here, both classifiers are trained on the same skewed data, while again the extractors differ. Therefore, this difference also isolates the effect of replacing the clean extractor with the skewed one. However, since the skewed data contain a systematic relationship between spurious features and labels, the classifiers face the same data-level incentive to use spurious features. In addition, the skewed feature extractor is incentivized to represent spurious features. Thus, this term isolates the counterfactual effect of introducing spurious features, including possible accompanying degradation of core features, e.g., due to representational capacity constraints. For this reason, we say that this term measures *spurious feature encoding* by the skewed feature extractor relative to the clean one.

In our analysis, we focus on feature properties and, thus, on the two metrics discussed above. The two remaining terms capture properties of the classifiers used on top and are mathematical complements of the previous notions. In the term $\mathrm{err}(c^s \circ f^s) - \mathrm{err}(c^c{}_{f^s} \circ f^s)$, both classifiers are adapted to the representations produced by the same extractor. Thus, each classifier's reliance on spurious features depends solely on the strength of statistical relationship between the spurious features and the labels in a dataset, given the skewed feature extractor. Hence, this term isolates the effect of training data on the reliance of the classifier on spurious features. We say that this term measures the *spurious feature amplification* in the skewed classifier (given the skewed feature extractor) due to the data skew. By the same logic, we say that the term $\mathrm{err}(c^s{}_{f^c} \circ f^c) - \mathrm{err}(c^c \circ f^c)$ measures *core feature underutilization* in the adapted classifier.

Counterfactual training algorithm To counterfactually interpret the metrics above, the training procedures for clean and skewed models and the adaptation procedures for the classifiers should be comparable with each other, i.e., they should control for all training factors except the presence of data skew. For the clean and skewed models, it can be achieved by using the same training procedures. However, for the adapted models, $c^c{}_{f^s} \circ f^s$ and $c^s{}_{f^c} \circ f^c$, this solution does not work since classifiers and feature extractors are trained on different data.

A potential solution to this adaptation problem is to use the standard empirical risk minimization procedure for adapting classifiers on top of fixed feature extractors. While this solution can work as a plausible counterfactual for linear classifiers, it fails to control for all training factors for deep classifiers. This is because deep models are often overparametrized and non-convex. Hence, for deep models, the final result depends not only on the achieved empirical risk, but also on initialization (Chizat et al., 2019), optimization algorithms (Gunasekar et al., 2018), optimizer hyperparameters (Keskar et al., 2017), explicit regularization (such as early stopping), and the prominence of different features at different stages of training (Panigrahi et al., 2025). Since explicitly controlling for all these factors is challenging, in Section 3.3, we propose a counterfactual training procedure that uses the same optimization protocol for network components.

3.2 Shortcut learning metrics

In the general case, we consider a feed-forward architecture consisting of m blocks of layers

$$f(\theta, \cdot) = f_{m-1}(\theta_{m-1}, \cdot) \circ f_{m-2}(\theta_{m-2}, \cdot) \circ f_0(\theta_0, \cdot). \tag{2}$$

Let θ_A be the weights of blocks $i \in A$, where $A \subseteq [m]$ and $[m] \coloneqq \{0, \ldots, m-1\}$. Define $i\!:\!j \coloneqq \{i, \ldots, j-1\}$ and $\mathrm{err}(\theta)$ as the error rate of architecture with weights θ on the clean test dataset. Consider two networks trained on clean and skewed data, resulting in weights θ^c and θ^s, respectively. Again, we interpret the increase in error rate $\mathrm{err}(\theta^s) - \mathrm{err}(\theta^c)$ as a measure of shortcut reliance. Our goal is to quantify the contributions of individual blocks to this metric.

We investigate what would happen if some intervention subset of blocks A of the clean (skewed) model were *counterfactually trained* on skewed (clean) data. Let $\theta^{c,A}$ be a model that shares blocks A with the clean model, but whose subset of blocks $[m] \setminus A$ was counterfactually trained on the skewed data (with $\theta^{s,A}$ defined analogously). As previously, we get decompositions

$$\mathrm{err}(\theta^s) - \mathrm{err}(\theta^c) = \mathrm{amp}_{[m]\setminus A} + \mathrm{fgt}_A = \mathrm{enc}_A + \mathrm{uut}_{[m]\setminus A}, \tag{3}$$

where[1]

[1]We expect all considered differences to be positive since skewness of the model progressively increases. But it is generally not guaranteed. For example, some layers might filter out a shortcut rule because it does not have perfect predictive power.

- $\text{fgt}_A \coloneqq \text{err}(\theta^{s,A}) - \text{err}(\theta^c)$ is the contribution of blocks A to *core feature forgetting*,
- $\text{enc}_A \coloneqq \text{err}(\theta^s) - \text{err}(\theta^{c,A})$ is the contribution of blocks A to *spurious feature encoding*,
- $\text{amp}_{[m]\setminus A} \coloneqq \text{err}(\theta^s) - \text{err}(\theta^{s,A})$ is the contribution of blocks $[m] \setminus A$ to *spurious feature amplification*,
- $\text{uut}_{[m]\setminus A} \coloneqq \text{err}(\theta^{c,A}) - \text{err}(\theta^c)$ is the contribution of blocks $[m] \setminus A$ to *core feature underutilization*.

As discussed previously, in our analysis, we focus on spurious feature encoding and core feature forgetting, with results on the localization of the other metrics being symmetrical. To allow comparison across datasets and architectures, we often consider *relative contributions* normalized by $\text{err}(\theta^s) - \text{err}(\theta^c)$ (see Section 4).

3.3 Counterfactual training algorithm

As previously argued, a key challenge in our approach is the design of a *counterfactual training* procedure that preserves learning mechanisms across all models. There are several critical factors: ensuring all corresponding blocks have *equal exposure to training data*, maintaining *consistent optimizer configurations and hyperparameters*, and allowing blocks to *progressively (i.e., during the course of training) adapt* to intermediate features since progressive adaptation invokes different learning mechanisms compared to static post-training (Allen-Zhu & Li, 2019; Panigrahi et al., 2025; Abbe et al., 2022).

We solve this challenge through a simultaneous training procedure described in Algorithm 1 (for brevity, we only consider training $\theta^{c,A}$ with stochastic gradient descent (SGD), but extensions to $\theta^{s,A}$ and other optimizers are straightforward). Here, the loss of a network with weights θ on a data batch B is denoted by $L(\theta, B)$. The procedure trains the anchor clean network θ^c and counterfactually intervened network $\theta^{c,A}$ in parallel. In each round, we sample a clean data batch B_t^c from D^c and skewed data batch $B_t^s = g(B_t^c)$ to update the models. Since blocks A are shared, we essentially only update $\theta^{c,A}_{[m]\setminus A}$ using skewed data during the backward pass through the network $\theta^{c,A}$.

Algorithm 1 Simultaneous training of networks

Initialize θ_0^c — anchor network weights
Initialize $\theta_0^{c,A} = \theta_0^c$ — intervened network weights
for $t = 1$ **to** T **do**
  Sample clean B_t^c and skewed $B_t^s = g(B_t^c)$ batches
  Update $\theta_t^{c,A} = \theta_{t-1}^{c,A} - \eta_t \nabla L(\theta_{t-1}^{c,A}, B_t^s)$
  Update $\theta_t^c = \theta_{t-1}^c - \eta_t \nabla L(\theta_{t-1}^c, B_t^c)$
  Synchronize shared blocks $\theta_{t,A}^{c,A} = \theta_{t,A}^c$
end for

Throughout this process, each counterfactually trained skewed block progressively adapts to the neighboring clean blocks to classify skewed data. By design, all blocks receive the same exposure to training data, with the only difference being the presence or the absence of a skew. The training algorithm remains consistent across all blocks, controlling for the optimizer's implicit biases and satisfying our desiderata.

4 Experiment details

We apply our methodology from Section 3 to study shortcut learning in vision models. This section outlines our experiment settings in detail.

Datasets and skews We use three datasets: CIFAR-10 (Krizhevsky, 2009), Waterbirds (Sagawa et al., 2020a), and CelebA (Liu et al., 2015) (with the standard task of hair color prediction). For CIFAR-10, we consider the watermark skew (see Figure 1), inspired by the MNIST—CIFAR-10 domino dataset (e.g., Shah et al., 2020; Trivedi et al., 2023), where we blend the upper-left corner of CIFAR images with class-correlated MNIST (Lecun et al., 1998) digits, encouraging the network to rely on the simple MNIST watermark. For

Waterbirds, we consider the standard background skew (Sagawa et al., 2020a), where we place bird images on class-correlated backgrounds, incentivizing the reliance on background cues. For CelebA, we consider the sampling skew, where we sample the skewed dataset introducing the standard spurious correlation between gender and hair color (Sagawa et al., 2020a), encouraging the reliance on the gender shortcut.

We generate skews using the following procedure. First, we create a clean dataset where the considered skew is not predictive. For CIFAR and Waterbirds, we simply match each image with a random MNIST digit image or background, respectively. For CelebA, we remove some images of blond females, non-blond males, and non-blond females to balance groups.[2] Then, we create a *fully skewed dataset* where the considered skew is perfectly predictive of the label. For CIFAR and Waterbirds, we replace a previously matched random image with an image corresponding to the image label if the class of the random image does not already correspond to the image label. For CelebA, we replace the images of blond males with blond females and the images of non-blond females with non-blond males, making the spurious correlation perfectly predictive. Finally, we create a skewed dataset, where each clean image is replaced with a corresponding fully skewed image with a certain frequency. We consider two frequencies: *near-universal* and *common* ($^{127}/_{128}$ and $^{15}/_{16}$, respectively). For CIFAR, we use watermarks of size 10×10 and two blending strengths, *strong* and *weak*, which equal to $^{3}/_{4}$ and $^{1}/_{4}$, respectively.

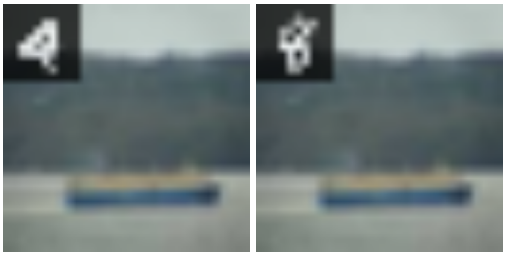

Figure 1: Clean (left) and skewed (right) CIFAR-10 image of class 8 with MNIST watermark

Models and optimizers We consider four architectures: VGG-11 (Simonyan & Zisserman, 2015), a typical convolutional neural network (CNN); ResNet-18 (He et al., 2016), a CNN with residual connections; DeiT-Ti (Touvron et al., 2021), a vision transformer (ViT); and ConvNeXt-T (Liu et al., 2022), a modernized CNN. Since all tasks have few classes, we use global average pooling instead of dense classification layers in VGG-11. We decompose each architecture into 6 blocks. The first block always corresponds to the initial convolutional layer, while the last block corresponds to the final linear layer. For the other blocks, the split is architecture-specific. For convolutional networks, each block corresponds to the input of a specific size. Specifically, for VGG, we use max-pool layers as block boundaries. For ResNet and ConvNeXt, we use convolutional layers with stride 2 as boundaries (see Figure 3 in He et al., 2016). As for DeiT, we simply divide the intermediate layers into four equal blocks. We consider SGD (Robbins & Monro, 1951) and AdamW (Loshchilov & Hutter, 2019) optimizers, and either train from scratch or fine-tune from ImageNet (Deng et al., 2009) initialization. (See Appendix C for details.)

Experiment scope We conduct two types of experiments. First, using sets $A = \{i\}$, we assess whether shortcuts could be localized in a single block by testing whether a *single block's contribution* could fully explain them, i.e.,

$$\exists i : \frac{\mathrm{fgt}_i}{\mathrm{err}(\theta^s) - \mathrm{err}(\theta^c)} \approx 1 \vee \frac{\mathrm{enc}_i}{\mathrm{err}(\theta^s) - \mathrm{err}(\theta^c)} \approx 1. \tag{4}$$

Second, to quantify layer-wise contributions, using sets $A = 0 : i$, we measure the *rate of increase in relative contributions of initial blocks* to forgetting and encoding,

$$\frac{\mathrm{fgt}_{0:i+1} - \mathrm{fgt}_{0:i}}{\mathrm{err}(\theta^s) - \mathrm{err}(\theta^c)} \quad \text{and} \quad \frac{\mathrm{enc}_{0:i+1} - \mathrm{enc}_{0:i}}{\mathrm{err}(\theta^s) - \mathrm{err}(\theta^c)}. \tag{5}$$

[2]The final number of blond and non-blond images equals the original number of blond females and non-blond males

5 Results

This section presents the results of our experiments for fine-tuned models (see Appendix B for training from scratch). In Section 5.1, we compute the localization metrics for individual and initial blocks. In Section 5.2, we analyze which factors contribute the most to the localization in the initial blocks. In Section 5.3, we investigate whether our localization metrics are predictive of the success of layer-wise training interventions.

To understand the extent of shortcut learning, Table 1 presents the clean test error of the models fine-tuned with AdamW on clean and skewed data for our datasets (SGD behavior is similar). We repeat each experiment 5 times and report the averaged values and their standard errors with Bessel's correction.

Table 1: Average clean test error rates (and their standard errors in parentheses) of clean and skewed models fine-tuned with AdamW for common (top part) and near-universal (bottom part) skews

	CIFAR-10 (weak)		CIFAR-10 (strong)		Waterbirds		CelebA	
Model	Clean	Skewed	Clean	Skewed	Clean	Skewed	Clean	Skewed
DeiT-Ti	3.2% (0.1%)	5.2% (0.1%)	3.4% (0.1%)	8.6% (0.1%)	2.0% (0.1%)	9.4% (0.4%)	5.8% (0.1%)	11.3% (0.2%)
ResNet-18	4.1% (0.1%)	7.2% (0.1%)	4.3% (0.1%)	11.5% (0.2%)	2.1% (0.1%)	8.5% (0.3%)	5.9% (0.1%)	11.3% (0.3%)
VGG-11	6.9% (0.1%)	17.1% (0.3%)	7.2% (0.1%)	26.3% (0.2%)	2.5% (0.1%)	10.7% (0.2%)	6.0% (0.1%)	11.0% (0.3%)
ConvNeXt-T	1.8% (0.1%)	3.3% (0.1%)	1.9% (0.1%)	5.1% (0.1%)	0.8% (0.1%)	3.7% (0.2%)	5.8% (0.1%)	11.4% (0.2%)
DeiT-Ti	3.2% (0.1%)	7.8% (0.2%)	3.4% (0.1%)	17.5% (0.3%)	2.0% (0.1%)	19.9% (0.6%)	5.8% (0.1%)	20.4% (0.8%)
ResNet-18	4.1% (0.1%)	11.3% (0.3%)	4.3% (0.1%)	25.3% (0.3%)	2.1% (0.1%)	18.2% (0.2%)	5.9% (0.1%)	19.4% (0.4%)
VGG-11	6.9% (0.1%)	29.2% (0.5%)	7.2% (0.1%)	52.4% (0.4%)	2.5% (0.1%)	25.3% (0.4%)	6.0% (0.1%)	21.3% (0.7%)
ConvNeXt-T	1.8% (0.1%)	5.4% (0.1%)	1.9% (0.1%)	11.6% (0.2%)	0.8% (0.1%)	10.9% (0.5%)	5.8% (0.1%)	20.8% (0.7%)

5.1 Localization of encoding and forgetting

Localization in a single block Table 2 presents the relative individual contributions of single blocks for the models fine-tuned with AdamW on the CIFAR-10 (strong) dataset with near-universal skew. First, no single block achieves 100% relative contribution to encoding or forgetting (according to the 5% critical value for one-sided t-test). Moreover, there is no individual block whose contribution is much (e.g., 10 times) larger than the contributions of other blocks. Second, the sum of individual contributions generally either significantly (according to the 5% critical value for the two-sided t-test) exceeds 100% (for encoding, with the exception of VGG-11 model) or does not reach 100% (for forgetting).

Our findings suggest that shortcut learning is not concentrated in any single block. The interactions between layers seem crucial for shortcut emergence. Similar patterns emerge across experimental settings (see additional results in Appendix A).[3]

Localization in initial blocks To account for layer interactions, we examined the localization in initial blocks (i.e., for $A = 0 : i$). Table 5 presents the results for models fine-tuned with AdamW on near-universal

[3]For single-block localization and fine-tuning, we discovered a rare possibility of model divergence (when the model achieves a higher error rate than the skewed model on clean data and a higher error rate than the clean model on skewed data). Divergences occurred in only 7 (out of 7040) intervened models (and only in the single-block setting). Such models were excluded from the analysis without impacting our conclusions.

Table 2: Average relative contributions (and SE) of single blocks to encoding (top) and forgetting (bottom) for fine-tuning with AdamW on CIFAR-10 (strong) with near-universal skew (blue $< 15\%$, red $\geq 15\%$)

Model	Bl. 0	Bl. 1	Bl. 2	Bl. 3	Bl. 4	Bl. 5
DeiT-Ti	0.1%	33.5%	42.9%	48.8%	25.4%	0.1%
	(1.7%)	(2.2%)	(2.0%)	(0.9%)	(2.3%)	(1.7%)
ResNet-18	2.4%	9.8%	16.9%	46.1%	66.0%	2.7%
	(1.2%)	(1.0%)	(1.4%)	(0.6%)	(0.5%)	(0.7%)
VGG-11	−0.5%	−0.2%	12.0%	53.0%	39.8%	4.1%
	(0.3%)	(0.2%)	(0.8%)	(1.7%)	(2.6%)	(0.5%)
ConvNeXt-T	−1.9%	4.5%	13.2%	76.4%	44.1%	0.1%
	(3.5%)	(1.3%)	(3.0%)	(1.2%)	(0.7%)	(1.6%)
DeiT-Ti	0.7%	1.0%	0.8%	0.4%	0.6%	−0.2%
	(0.5%)	(0.4%)	(0.6%)	(0.4%)	(1.0%)	(0.4%)
ResNet-18	0.2%	0.6%	1.8%	2.7%	5.1%	0.3%
	(0.4%)	(0.3%)	(0.4%)	(0.6%)	(0.2%)	(0.2%)
VGG-11	−0.1%	0.2%	0.8%	10.0%	9.7%	0.3%
	(0.3%)	(0.3%)	(0.3%)	(1.4%)	(1.7%)	(0.2%)
ConvNeXt-T	−0.4%	1.0%	0.8%	1.6%	1.0%	−0.6%
	(0.9%)	(1.0%)	(0.8%)	(0.7%)	(0.8%)	(0.6%)

skews. To make the results comparable, we report the rate of increase in relative contributions. As expected, encoding and forgetting generally increase with the number of layers involved. All architectures first encode spurious features and subsequently forget core features. The last layer plays a major role in feature forgetting, while the first layer has modest contributions.[4]

Figure 2 presents the average contributions of blocks across experiments. The first two blocks modestly contribute to shortcut learning, the middle blocks mostly contribute to shortcut encoding, the penultimate block contributes to both encoding and forgetting, and the last layer majorly contributes to forgetting.

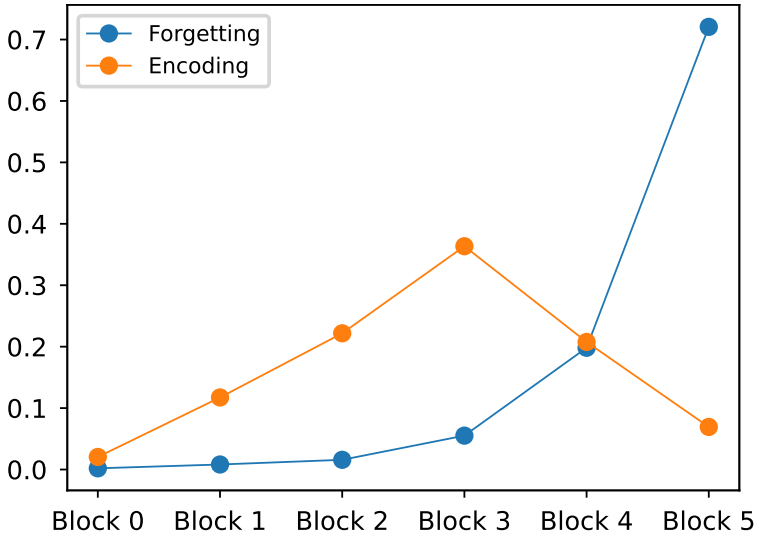

Figure 2: Average relative encoding and forgetting rates

[4]For training from scratch, the first layer starts to engage in spurious feature encoding, see Table 14 in Appendix.

These results indicate that shortcut learning indeed consists of two processes that occur in different layers. First, the network encodes spurious features; then, due to predictive spurious features, it progressively forgets core features, leading to a shortcut classification rule.

5.2 Differences in relative contributions

Main explanatory factors Table 3 reports the fraction of the total variance of the increase rate of relative encoding and forgetting explained by different factors. Dataset and model architecture are the most predictive factors for encoding, while skew frequency and optimizer choice are the most predictive for forgetting. Together, dataset and model factors explain 83.8% of variance in encoding localization, while skew frequency and optimizer factors explain 57.0% of variance in forgetting localization.

Table 3: Variance explained in relative encoding (top) and forgetting (bottom) rate by different factors

Dataset	Skew freq.	Model	Optimizer
45.7%	0.4%	26.3%	3.3%
14.7%	21.0%	9.1%	33.5%

These findings suggest that encoding localization is primarily driven by the skew's semantic properties and the architecture. At the same time, forgetting appears to be primarily influenced by the skew's predictive power (through the dataset and skew frequency) and the optimizer. We further explore these factors below.

Differences in encoding Figure 3 (top part) shows the average increase rate in relative encoding for the specified groups. As we can see, sampling skew is mostly encoded in the penultimate block, watermark skew is encoded in the middle blocks, and background skew is encoded in both middle and penultimate blocks. Also, note that sampling skew corresponds to a non-localizable complex feature (gender), while the watermark and background skews are localizable and relatively easier. Thus, we can speculate that the middle layers operate on more local features, while the penultimate block operates on higher-level global features. This suggestion echoes the recent literature on feature encoding in CNNs (Fel et al., 2024). Additionally, we observe that encoding occurs earlier in DeiT-Ti compared to CNN architectures.

Differences in forgetting Figure 3 (bottom part) shows the average increase rate in relative forgetting per specified groups. We can see that forgetting due to near-universal skews is more concentrated in the last layer than that induced by common skews. This fact, along with the major role of the last layer in feature forgetting, suggests that the last layer seeks to select more predictive features for classification and filters out others, mirroring the findings of Kirichenko et al. (2023). At the same time, the non-trivial role of the penultimate block in both encoding and forgetting, along with a more modest response to skew's predictive power, suggests that the penultimate block does not actively filter out less predictive features. Instead, we could speculate that, in the penultimate block, spurious features "squeeze out" core features due to capacity constraints. Additionally, we note that SGD prioritizes forgetting in the last layer, whereas AdamW distributes it between the last layer and the penultimate block.

5.3 Layer-wise interventions

This section explores whether our localization metrics can predict the success of layer-wise interventions in shortcut mitigation. We retrained DeiT-Ti, ResNet-18, and VGG-11 on skewed data, using near-universal skews in Waterbirds, CelebA, and CIFAR-10 (strong). We consider four retraining interventions, where we modify the hyperparameters of the optimizer layer-wise: 1–2) increasing or decreasing LR (learning rate) by a factor of 3, and 3–4) increasing or decreasing WD (weight decay) by a factor of 10. We applied these interventions to individual blocks. Additionally, we conduct an experiment where we train the last layer and then the whole network for a short time, and then freeze all blocks except one during fine-tuning.

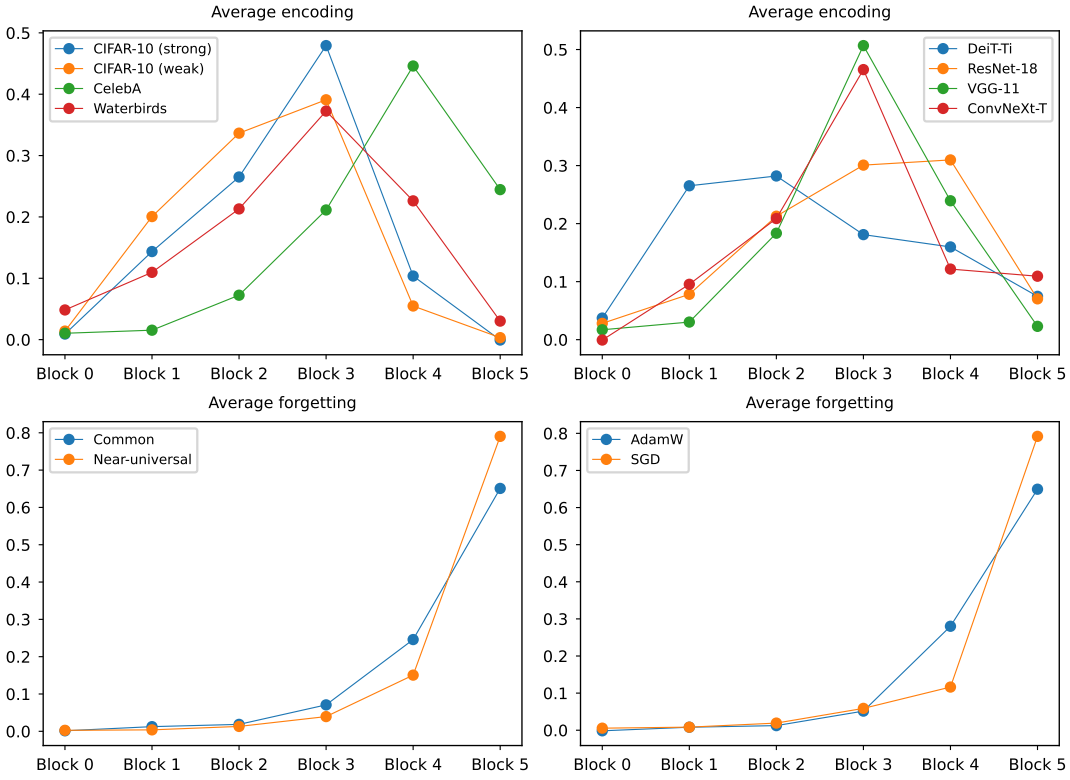

Figure 3: Average encoding and forgetting rates per group

We focus on shortcut mitigation as the main variable of interest and define it as

$$\text{mitigation}_i = \frac{\text{err}(\theta^s) - \text{err}(\theta_i^{\text{int}})}{\text{err}(\theta^s) - \text{err}(\theta^c)}, \tag{6}$$

where θ_i^{int} is the model intervened at block i. We regress the shortcut mitigation metric on encoding and forgetting metrics, their interaction (i.e., their product), their squares and a last-layer dummy variable.

Table 4 presents the results for the relevance of our metrics (the F-statistic (Greene, 2003) for the joint significance of the five metric-related coefficients) and overall predictive power of the regression (the R^2).[5] Our localization metrics are predictive of success for LR and freezing interventions (at the 5% significance level). This finding suggests that our localization metrics capture relevant information and could inform shortcut mitigations.

Table 4: Predictive power of localization metrics

	LR↑	LR↓	WD↑	WD↓	Freeze
F-Stat	4.78	4.42	0.46	0.68	7.88
R^2	0.25	0.31	0.02	0.04	0.11
N	108	108	108	108	108

[5]See Table 6 in Appendix for regression coefficients.

Table 5: Average increase rate in relative contributions (and SE) of the initial blocks to encoding (top) and forgetting (bottom) for fine-tuning with AdamW on near-universal skews (blue $< 15\%$, red $\geq 15\%$)

Dataset	Model	Bl. 0	Bl. 1	Bl. 2	Bl. 3	Bl. 4	Bl. 5
CIFAR-10 (strong)	DeiT-Ti	1.1%	40.3%	39.5%	15.1%	4.3%	0.1%
		(3.1%)	(2.0%)	(1.8%)	(0.7%)	(0.4%)	(0.1%)
	ResNet-18	2.4%	10.5%	26.1%	45.6%	15.7%	0.1%
		(1.2%)	(1.0%)	(1.4%)	(0.8%)	(0.6%)	(0.1%)
	VGG-11	−0.5%	1.5%	17.8%	72.3%	9.1%	0.0%
		(0.3%)	(0.2%)	(0.3%)	(0.2%)	(0.3%)	(0.1%)
	ConvNeXt-T	−1.8%	6.7%	24.6%	67.6%	3.3%	−0.1%
		(3.8%)	(5.3%)	(2.1%)	(1.9%)	(0.2%)	(0.2%)
Waterbirds	DeiT-Ti	1.2%	19.1%	30.3%	24.6%	25.0%	0.0%
		(2.4%)	(2.5%)	(2.8%)	(1.8%)	(1.1%)	(0.1%)
	ResNet-18	6.7%	11.7%	18.9%	27.2%	35.4%	0.4%
		(1.5%)	(1.1%)	(1.0%)	(0.9%)	(1.0%)	(0.3%)
	VGG-11	4.5%	9.8%	18.9%	51.6%	14.9%	0.6%
		(0.9%)	(2.8%)	(1.8%)	(1.8%)	(0.9%)	(0.2%)
	ConvNeXt-T	7.2%	2.6%	15.0%	62.4%	13.0%	0.2%
		(2.2%)	(4.8%)	(3.6%)	(2.2%)	(1.6%)	(0.2%)
CelebA	DeiT-Ti	2.0%	5.1%	8.7%	18.5%	58.9%	7.2%
		(0.7%)	(2.1%)	(1.4%)	(0.6%)	(2.1%)	(0.6%)
	ResNet-18	0.7%	−3.2%	9.2%	23.0%	64.5%	6.2%
		(1.7%)	(1.4%)	(0.7%)	(1.8%)	(1.7%)	(1.4%)
	VGG-11	4.1%	1.5%	7.1%	27.7%	60.0%	0.1%
		(0.6%)	(0.6%)	(1.3%)	(1.9%)	(2.3%)	(0.4%)
	ConvNeXt-T	−0.7%	3.0%	3.0%	18.1%	59.4%	17.5%
		(1.7%)	(1.2%)	(1.7%)	(2.2%)	(2.0%)	(1.0%)
CIFAR-10 (strong)	DeiT-Ti	0.6%	1.1%	1.9%	5.2%	11.0%	80.6%
		(0.5%)	(0.5%)	(0.4%)	(0.2%)	(0.5%)	(0.8%)
	ResNet-18	0.2%	0.9%	1.8%	2.0%	19.3%	76.1%
		(0.4%)	(0.3%)	(0.4%)	(0.4%)	(0.7%)	(0.5%)
	VGG-11	−0.1%	0.2%	0.7%	2.2%	21.7%	75.7%
		(0.3%)	(0.2%)	(0.2%)	(0.3%)	(0.3%)	(0.5%)
	ConvNeXt-T	−0.2%	1.6%	−0.4%	5.1%	13.5%	80.7%
		(0.8%)	(0.6%)	(0.3%)	(0.6%)	(0.9%)	(0.9%)
Waterbirds	DeiT-Ti	0.7%	0.5%	1.9%	5.6%	23.0%	68.6%
		(0.4%)	(0.2%)	(0.5%)	(0.4%)	(0.9%)	(0.9%)
	ResNet-18	0.6%	1.1%	0.5%	2.5%	28.6%	67.0%
		(0.4%)	(0.2%)	(0.4%)	(0.6%)	(3.0%)	(3.2%)
	VGG-11	0.3%	0.2%	−0.2%	2.8%	29.8%	67.3%
		(0.3%)	(0.3%)	(0.5%)	(0.7%)	(1.0%)	(1.0%)
	ConvNeXt-T	0.6%	0.5%	0.5%	2.2%	16.6%	79.8%
		(0.3%)	(0.6%)	(0.3%)	(0.2%)	(0.6%)	(0.6%)
CelebA	DeiT-Ti	0.0%	0.4%	2.2%	2.1%	10.9%	84.7%
		(0.1%)	(0.4%)	(0.5%)	(0.5%)	(0.7%)	(1.1%)
	ResNet-18	−0.2%	0.1%	1.0%	3.3%	15.9%	80.2%
		(0.4%)	(0.2%)	(0.4%)	(0.7%)	(1.0%)	(0.7%)
	VGG-11	0.1%	0.4%	1.9%	4.6%	26.5%	66.9%
		(0.3%)	(0.2%)	(0.2%)	(0.8%)	(2.4%)	(1.7%)
	ConvNeXt-T	−0.3%	−0.0%	−0.0%	2.0%	10.7%	87.9%
		(0.3%)	(0.3%)	(0.2%)	(0.4%)	(0.6%)	(1.1%)

6 Discussion

We analyzed the localization of shortcuts in deep models. We qualified the layer-wise contributions of layers to shortcut learning using our counterfactual training method and two metrics: spurious feature encoding and core feature forgetting. Our findings demonstrate that neither encoding nor forgetting is localized in any single layer within vision models. The interactions between layers play a crucial role in shortcut formation. Earlier blocks typically facilitate spurious feature encoding, while later blocks are responsible for core feature forgetting.

By examining the axes of variation in our metrics, we found that dataset and model architecture play a crucial role in encoding localization. At the same time, skew frequency and optimizer are important for forgetting. Additionally, our localization metrics are predictive of the success of certain layer-wise interventions. Together, these results suggest that the success of layer-wise interventions can vary considerably across learning settings, underscoring the difficulty of designing layer-wise mitigation strategies.

Future work To highlight one potential avenue for future work, note that our observations suggest a trade-off between feature extractor adaptability to additional predictive features and robustness to shortcuts. Feature extractors trained on clean data provide greater robustness to shortcuts by compelling the final classifier layers to rely on core features. However, models with clean feature extractors have a higher error rate on fully skewed data (i.e., on data where the spurious features are fully predictive; see Table 7 in Appendix). This behavior suggests that simply aiming for high validation accuracy might be an inappropriate training strategy because it could encourage shortcut learning. On the other hand, since most of the forgetting occurs in the last layers, training on skewed data and then fixing those layers offers a reasonable learning strategy. However, this strategy results in a fragile feature extractor that encodes spurious features. It would be interesting to investigate this phenomenon in greater depth, as this can help to design feature extractors that generalize across many tasks.

References

Emmanuel Abbe, Enric Boix Adsera, and Theodor Misiakiewicz. The merged-staircase property: a necessary and nearly sufficient condition for SGD learning of sparse functions on two-layer neural networks. In Po-Ling Loh and Maxim Raginsky (eds.), *Proceedings of Thirty Fifth Conference on Learning Theory*, volume 178 of *Proceedings of Machine Learning Research*, pp. 4782–4887. PMLR, 02–05 Jul 2022.

Zeyuan Allen-Zhu and Yuanzhi Li. What Can ResNet Learn Efficiently, Going Beyond Kernels? In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant Risk Minimization, 2020. arXiv:1907.02893 [stat.ML].

Marc-Etienne Brunet, Ashton Anderson, and Richard Zemel. Implications of Model Indeterminacy for Explanations of Automated Decisions. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 7810–7823. Curran Associates, Inc., 2022.

Kamalika Chaudhuri, Kartik Ahuja, Martin Arjovsky, and David Lopez-Paz. Why does Throwing Away Data Improve Worst-Group Error? In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 4144–4188. PMLR, 23–29 Jul 2023.

Lénaïc Chizat, Edouard Oyallon, and Francis Bach. On Lazy Training in Differentiable Programming. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

Thomas Fel, Louis Béthune, Andrew Kyle Lampinen, Thomas Serre, and Katherine Hermann. Understanding Visual Feature Reliance through the Lens of Complexity. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 69888–69924. Curran Associates, Inc., 2024.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.

William H Greene. Econometric analysis. *Pretence Hall*, 2003.

Suriya Gunasekar, Jason Lee, Daniel Soudry, and Nathan Srebro. Characterizing Implicit Bias in Terms of Optimization Geometry. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1832–1841. PMLR, 10–15 Jul 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2016.

Katherine Hermann and Andrew Lampinen. What shapes feature representations? exploring datasets, architectures, and training. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 9995–10006. Curran Associates, Inc., 2020.

Minyoung Huh, Hossein Mobahi, Richard Zhang, Brian Cheung, Pulkit Agrawal, and Phillip Isola. The Low-Rank Simplicity Bias in Deep Networks. *Transactions on Machine Learning Research*, 2023.

Md Amirul Islam, Matthew Kowal, Patrick Esser, Sen Jia, Björn Ommer, Konstantinos G. Derpanis, and Neil Bruce. Shape or Texture: Understanding Discriminative Features in CNNs. In *International Conference on Learning Representations*, 2021.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. In *International Conference on Learning Representations*, 2017.

Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. Last Layer Re-Training is Sufficient for Robustness to Spurious Correlations. In *International Conference on Learning Representations*, 2023.

Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images, 2009.

Ananya Kumar, Aditi Raghunathan, Robbie Matthew Jones, Tengyu Ma, and Percy Liang. Fine-Tuning can Distort Pretrained Features and Underperform Out-of-Distribution. In *International Conference on Learning Representations*, 2022.

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

Yoonho Lee, Annie S Chen, Fahim Tajwar, Ananya Kumar, Huaxiu Yao, Percy Liang, and Chelsea Finn. Surgical Fine-Tuning Improves Adaptation to Distribution Shifts. In *International Conference on Learning Representations*, 2023.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A ConvNet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11976–11986, June 2022.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep Learning Face Attributes in the Wild. In *Proceedings of the IEEE International Conference on Computer Vision*, December 2015.

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*, 2019.

Pratyush Maini, Michael Curtis Mozer, Hanie Sedghi, Zachary Chase Lipton, J Zico Kolter, and Chiyuan Zhang. Can Neural Network Memorization Be Localized? In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 23536–23557. PMLR, 23–29 Jul 2023.

Vaishnavh Nagarajan, Anders Andreassen, and Behnam Neyshabur. Understanding the failure modes of out-of-distribution generalization. In *International Conference on Learning Representations*, 2021.

Abhishek Panigrahi, Bingbin Liu, Sadhika Malladi, Andrej Risteski, and Surbhi Goel. Progressive distillation induces an implicit curriculum. In *The Thirteenth International Conference on Learning Representations*, 2025.

Aahlad Puli, Lily Zhang, Yoav Wald, and Rajesh Ranganath. Don't blame Dataset Shift! Shortcut Learning due to Gradients and Cross Entropy. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 71874–71910. Curran Associates, Inc., 2023.

Herbert Robbins and Sutton Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. Distributionally Robust Neural Networks. In *International Conference on Learning Representations*, 2020a.

Shiori Sagawa, Aditi Raghunathan, Pang Wei Koh, and Percy Liang. An Investigation of Why Overparameterization Exacerbates Spurious Correlations. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8346–8356. PMLR, 13–18 Jul 2020b.

Luca Scimeca, Seong Joon Oh, Sanghyuk Chun, Michael Poli, and Sangdoo Yun. Which Shortcut Cues Will DNNs Choose? A Study from the Parameter-Space Perspective. In *International Conference on Learning Representations*, 2022.

Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. The Pitfalls of Simplicity Bias in Neural Networks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 9573–9585. Curran Associates, Inc., 2020.

Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*, 2015.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, 2014. arXiv:1312.6034 [cs.CV].

TorchVision maintainers and contributors. TorchVision: PyTorch's Computer Vision library, 2016.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers & distillation through attention. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 10347–10357. PMLR, 18–24 Jul 2021.

Puja Trivedi, Danai Koutra, and Jayaraman J. Thiagarajan. A Closer Look at Model Adaptation using Feature Distortion and Simplicity Bias. In *International Conference on Learning Representations*, 2023.

Nikita Tsoy and Nikola Konstantinov. Simplicity Bias of Two-Layer Networks beyond Linearly Separable Data. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 48728–48767. PMLR, 21–27 Jul 2024.

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

Shunxin Wang, Raymond Veldhuis, Christoph Brune, and Nicola Strisciuglio. What do neural networks learn in image classification? a frequency shortcut perspective. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1433–1442, October 2023.

Ross Wightman. PyTorch Image Models, 2019.

Chiyuan Zhang, Samy Bengio, and Yoram Singer. Are All Layers Created Equal? *Journal of Machine Learning Research*, 23(67):1–28, 2022.

Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million Image Database for Scene Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

Haoran Zhu, Boyuan Chen, and Carter Yang. Understanding Why ViT Trains Badly on Small Datasets: An Intuitive Perspective, 2023. arXiv:2302.03751 [cs.CV].

Appendices

- Appendix A provides additional results of experiments for fine-tuned models.
- Appendix B analyzes training from scratch.
- Appendix C provides training details.
- Appendix D lists licences of used assets.

A Additional results for fine-tuned models

Layer-wise interventions Table 6 present the full version of Table 4. As we can see, for the LR↑ and Freeze interventions, the success is positively correlated with forgetting and negatively correlated with encoding metrics. At the same time, the mitigation outcome is negatively correlated with the square of the forgetting metric and the interaction and positively correlated with the square of the encoding metric. Correspondingly, the coefficients the LR↓ intervention have the opposite signs. For the WD interventions, all coefficients of the regression are small and statistically insignificant.

Table 6: Predictive power of localization metrics with regression coefficients (and their standard errors)

	LR↑	LR↓	WD↑	WD↓	Freeze
Enc	−0.13	0.11	0.03	0.06	−1.04
	(0.09)	(0.07)	(0.06)	(0.06)	(0.96)
Fgt	0.49	−0.48	−0.01	−0.02	3.55
	(0.11)	(0.12)	(0.06)	(0.05)	(0.69)
Enc × Fgt	−0.08	0.07	−0.03	−0.11	−2.47
	(0.16)	(0.15)	(0.09)	(0.08)	(1.17)
Enc^2	0.05	−0.07	−0.07	−0.10	1.51
	(0.12)	(0.11)	(0.08)	(0.08)	(1.27)
Fgt^2	−0.33	0.32	0.04	0.07	−3.23
	(0.09)	(0.10)	(0.08)	(0.07)	(0.89)
Const	−0.01	0.00	−0.00	−0.00	−0.22
	(0.01)	(0.01)	(0.01)	(0.01)	(0.13)
Last	−0.14	0.14	−0.02	−0.02	−0.61
	(0.07)	(0.07)	(0.04)	(0.04)	(0.43)
F-Stat	4.78	4.42	0.46	0.68	7.88
R^2	0.25	0.31	0.02	0.04	0.11
N	108	108	108	108	108

Error rates on fully skewed dataset Table 7 presents error rates on fully skewed data for models $\theta^c_{0:i}$ fine-tuned with AdamW on CIFAR-10 (strong) with near-universal skew. As we can see, the error rates increase with the number of initial clean blocks.

Localization in a single block Table 8 follows Table 2 for the models fine-tuned with AdamW on CelebA and Waterbirds with near-universal skew. Similarly to the CIFAR-10 results, there does not exist an individual layer fully responsible for shortcut learning. As previously, the sum of individual contributions to forgetting does not reach 100%, which suggest that forgetting can not be explained by the sum of individual contributions. In contrast to the previous results, the sum of individual contributions to encoding does not reach 100% for CelebA and approximately equals 100% for Waterbirds. These results again suggest that the sum of individual contributions can not reliably explain spurious feature encoding. While this approach gives plausible results for the Waterbirds dataset, it fails for the CIFAR-10 and CelebA datasets.

Table 7: Average skewed error rates (and SE) for AdamW fine-tuning on CIFAR-10 (strong) with the near-universal skew

Model	Skewed	0 : 1	0 : 2	0 : 3	0 : 4	0 : 5	Clean
DeiT-Ti	0.3% (0.1%)	0.3% (0.1%)	0.5% (0.1%)	1.3% (0.1%)	2.3% (0.1%)	3.4% (0.1%)	3.4% (0.1%)
ResNet-18	0.3% (0.1%)	0.3% (0.1%)	0.3% (0.1%)	0.4% (0.1%)	1.3% (0.1%)	4.2% (0.1%)	4.2% (0.1%)
VGG-11	0.3% (0.1%)	0.3% (0.1%)	0.3% (0.1%)	0.3% (0.1%)	2.4% (0.1%)	7.2% (0.1%)	7.2% (0.1%)

Interestingly, Block 4 of the VGG-11 model exhibits a strong negative contribution to encoding. To understand this behavior, we examined the individual contributions to encoding for the same models trained on common skew in Table 9 and the error rates of these models in Table 10. Similarly to the near-universal skew, Block 4 of the VGG-11 model exhibits a strong negative contribution to encoding. Also, all intervened models achieve a small error rate on the fully skewed dataset. A plausible explanation of this behavior is the following. Block 4 in the clean model is not well suited for spurious feature encoding. Due to regularization (i.e., implicit biases of optimizer and weight decay), the complement to this block can not overcome this restriction and starts to mix core features with spurious features, which corrupts the core features and leads to a significant accuracy drop on the clean dataset. This example again suggests that shortcut learning crucially depends on the interactions of different layers within the network.

Table 8: Average relative contributions (and standard errors) of single blocks to encoding (first and third sub-part) and forgetting (second and fourth sub-parts) for models fine-tuned with AdamW on CelebA (top part) and Waterbirds (bottom part) with near-universal skew

Model	Bl. 0	Bl. 1	Bl. 2	Bl. 3	Bl. 4	Bl. 5
DeiT-Ti	2.0%	3.7%	6.6%	−0.1%	2.1%	0.7%
	(0.7%)	(1.9%)	(1.9%)	(1.5%)	(1.3%)	(0.6%)
ResNet-18	0.7%	−1.4%	0.7%	−11.7%	12.1%	−0.1%
	(1.7%)	(5.9%)	(3.1%)	(3.2%)	(5.6%)	(1.0%)
VGG-11	4.1%	3.2%	4.9%	5.0%	−73.3%	−4.3%
	(0.6%)	(1.0%)	(2.4%)	(6.8%)	(10.6%)	(0.7%)
ConvNeXt-T	0.2%	2.6%	2.2%	9.8%	−0.4%	−2.6%
	(1.8%)	(1.7%)	(2.3%)	(3.6%)	(2.6%)	(1.1%)
DeiT-Ti	0.0%	0.2%	0.9%	0.5%	0.2%	0.0%
	(0.1%)	(0.2%)	(0.2%)	(0.2%)	(0.3%)	(0.2%)
ResNet-18	−0.2%	0.8%	0.4%	2.3%	23.8%	−0.1%
	(0.4%)	(0.5%)	(0.3%)	(0.4%)	(13.6%)	(0.2%)
VGG-11	0.1%	0.3%	1.7%	7.9%	0.6%	−0.1%
	(0.3%)	(0.3%)	(0.7%)	(2.2%)	(0.7%)	(0.3%)
ConvNeXt-T	−0.0%	−0.4%	−0.2%	1.4%	0.1%	−0.0%
	(0.3%)	(0.2%)	(0.4%)	(0.7%)	(0.5%)	(0.2%)
DeiT-Ti	1.2%	16.7%	32.4%	31.2%	14.8%	3.2%
	(2.4%)	(2.1%)	(4.8%)	(2.4%)	(2.6%)	(1.5%)
ResNet-18	6.7%	12.4%	24.8%	40.7%	17.0%	1.4%
	(1.5%)	(1.8%)	(1.9%)	(3.5%)	(7.0%)	(0.5%)
VGG-11	4.5%	10.2%	31.3%	64.3%	18.9%	2.0%
	(0.9%)	(2.9%)	(0.6%)	(2.3%)	(4.7%)	(0.5%)
ConvNeXt-T	−3.1%	2.4%	12.7%	77.7%	51.3%	1.7%
	(5.3%)	(2.0%)	(4.7%)	(3.1%)	(1.3%)	(4.7%)
DeiT-Ti	0.7%	0.7%	1.1%	1.8%	0.9%	0.1%
	(0.4%)	(0.4%)	(0.3%)	(0.4%)	(0.4%)	(0.4%)
ResNet-18	0.6%	0.5%	0.7%	1.1%	20.7%	0.1%
	(0.4%)	(0.4%)	(0.2%)	(0.5%)	(3.7%)	(0.3%)
VGG-11	0.3%	0.2%	1.1%	7.0%	1.4%	0.2%
	(0.3%)	(0.2%)	(0.6%)	(1.1%)	(0.2%)	(0.1%)
ConvNeXt-T	1.5%	1.2%	1.1%	1.8%	1.5%	0.6%
	(1.0%)	(0.8%)	(0.6%)	(0.4%)	(0.7%)	(0.3%)

Table 9: Average relative contributions (and standard errors) of single blocks to encoding for models fine-tuned with AdamW on CelebA with common skew

Model	Bl. 0	Bl. 1	Bl. 2	Bl. 3	Bl. 4	Bl. 5
VGG-11	4.1%	2.2%	−3.6%	−6.1%	−56.7%	−1.9%
	(0.9%)	(1.6%)	(2.1%)	(7.6%)	(12.7%)	(0.7%)

Table 10: Average test error rates (and standard errors) on the clean (first and third row) and fully skewed (second and fourth row) CelebA dataset of clean models intervened with sets $A = [m] \setminus \{i\}$ for VGG-11 architecture fine-tuned with AdamW on near-universal (top) and common (bottom) skews

Skewed	$\{-0\}$	$\{-1\}$	$\{-2\}$	$\{-3\}$	$\{-4\}$	$\{-5\}$	Clean
21.3%	20.7%	20.8%	20.6%	20.5%	32.5%	22.0%	6.0%
(0.7%)	(0.6%)	(0.6%)	(0.9%)	(0.9%)	(1.9%)	(0.6%)	(0.1%)
0.31%	0.31%	0.31%	0.35%	0.50%	0.49%	0.31%	5.09%
(0.01%)	(0.02%)	(0.02%)	(0.02%)	(0.06%)	(0.05%)	(0.01%)	(0.04%)
11.0%	10.8%	10.9%	11.2%	11.2%	13.7%	11.1%	6.0%
(0.3%)	(0.3%)	(0.4%)	(0.4%)	(0.2%)	(0.7%)	(0.4%)	(0.1%)
0.64%	0.67%	0.66%	0.67%	0.75%	0.56%	0.63%	5.09%
(0.04%)	(0.04%)	(0.05%)	(0.06%)	(0.05%)	(0.01%)	(0.04%)	(0.04%)

B Training from scratch

This section presents the general trends for models trained from scratch observed in our experiments. In this section, we did not train ConvNeXt-T model from scratch due to computational constraints. Also, note that we did not manage to train the DeiT-Ti architecture on the Waterbirds dataset from scratch since the size of the Waterbirds dataset is not sufficient to train a "data hungry" (Zhu et al., 2023) transformer architecture using the standard training methods. Thus, experiments for this pair are omitted. As previously, we first report the error rates achieved by models on different datasets in Table 11. As we can see, shortcut learning is exacerbated for models trained from scratch. Also, we see that DeiT models have significantly higher error rates on CIFAR-10 because, again, ViT architectures are more "data hungry". Also, we can see that AdamW achieves better error rates compared to SGD for ViT architectures, while the opposite is true for CNN architectures.

Table 11: Average clean test error rates (and their standard errors in parenthesis) of clean and skewed models trained from scratch with AdamW (top part) and SGD (bottom part) on common (first and third sub-parts) and near-universal (second and fourth sub-parts) skews

	CIFAR-10 (weak)		CIFAR-10 (strong)		Waterbirds		CelebA	
Model	Clean	Skewed	Clean	Skewed	Clean	Skewed	Clean	Skewed
DeiT-Ti	18.7%	20.8%	18.9%	41.8%	–	–	7.1%	13.3%
	(0.2%)	(0.3%)	(0.2%)	(0.4%)	–	–	(0.1%)	(0.1%)
ResNet-18	6.7%	19.8%	7.0%	25.6%	7.1%	26.9%	6.3%	11.3%
	(0.2%)	(0.2%)	(0.1%)	(0.2%)	(0.2%)	(0.5%)	(0.1%)	(0.4%)
VGG-11	6.7%	24.2%	7.0%	28.7%	4.9%	25.0%	6.3%	11.2%
	(0.1%)	(0.3%)	(0.1%)	(0.3%)	(0.2%)	(0.6%)	(0.1%)	(0.4%)
DeiT-Ti	18.4%	25.2%	18.7%	63.3%	–	–	7.1%	22.3%
	(0.2%)	(0.6%)	(0.1%)	(0.3%)	–	–	(0.1%)	(0.3%)
ResNet-18	6.7%	35.1%	7.0%	48.4%	7.1%	36.7%	6.3%	21.4%
	(0.2%)	(0.4%)	(0.1%)	(0.3%)	(0.2%)	(0.4%)	(0.1%)	(0.6%)
VGG-11	6.7%	46.1%	7.0%	57.5%	4.9%	35.2%	6.3%	20.5%
	(0.1%)	(0.5%)	(0.1%)	(0.4%)	(0.2%)	(0.5%)	(0.1%)	(0.3%)
DeiT-Ti	27.5%	28.2%	27.6%	44.1%	–	–	8.7%	12.7%
	(0.3%)	(0.2%)	(0.2%)	(0.3%)	–	–	(0.2%)	(0.2%)
ResNet-18	5.6%	18.3%	5.9%	24.5%	6.5%	27.3%	6.3%	11.4%
	(0.1%)	(0.1%)	(0.1%)	(0.2%)	(0.2%)	(0.8%)	(0.1%)	(0.2%)
VGG-11	6.4%	23.1%	6.8%	28.6%	4.9%	25.4%	6.3%	11.2%
	(0.1%)	(0.1%)	(0.2%)	(0.2%)	(0.1%)	(0.5%)	(0.1%)	(0.2%)
DeiT-Ti	27.8%	28.7%	27.4%	59.1%	–	–	8.8%	14.9%
	(0.3%)	(0.3%)	(0.3%)	(0.3%)	–	–	(0.2%)	(0.3%)
ResNet-18	5.6%	32.6%	5.9%	47.9%	6.5%	36.5%	6.3%	22.0%
	(0.1%)	(0.4%)	(0.1%)	(0.3%)	(0.2%)	(0.4%)	(0.1%)	(0.4%)
VGG-11	6.4%	43.8%	6.8%	57.0%	4.9%	35.2%	6.3%	19.9%
	(0.1%)	(0.5%)	(0.2%)	(0.5%)	(0.1%)	(0.6%)	(0.1%)	(0.2%)

Localization in a single block Table 12 replicates Table 2 for the models trained from scratch. In this experiment, we observed divergence in some intervened VGG-11 models: contributions of the corresponding models are not included in the mean calculation.[6] Generally, we observe trends similar to those previously observed with the fine-tuned models.

[6]Similarly to the fine-tuning case, we did not observe divergence in the initial blocks setting. In the single block setting, we observed divergence in 227 out of 2640 models.

Table 12: Average relative contributions (and standard errors) of single blocks to encoding (top) and forgetting (bottom) for models trained from scratch with AdamW on CIFAR-10 (strong) with near-universal skew

Model	Bl. 0	Bl. 1	Bl. 2	Bl. 3	Bl. 4	Bl. 5
DeiT-Ti	24.1%	9.0%	13.7%	14.2%	11.6%	−1.1%
	(1.0%)	(0.6%)	(0.9%)	(1.1%)	(1.1%)	(0.8%)
ResNet-18	0.3%	−17.8%	−23.5%	−0.6%	19.3%	0.6%
	(1.1%)	(6.3%)	(9.1%)	(4.7%)	(2.1%)	(1.1%)
VGG-11	3.4%	–	−24.2%	25.9%	16.9%	0.0%
	(0.7%)	–	–	(2.8%)	(3.4%)	(0.6%)
DeiT-Ti	−0.4%	−1.4%	0.7%	0.3%	0.5%	−0.2%
	(0.2%)	(0.5%)	(0.5%)	(0.4%)	(0.4%)	(0.4%)
ResNet-18	0.8%	2.7%	21.8%	18.7%	23.3%	−0.1%
	(0.3%)	(0.4%)	(6.2%)	(2.2%)	(2.1%)	(0.3%)
VGG-11	0.7%	61.2%	63.7%	57.1%	–	1.4%
	(0.4%)	(21.7%)	(28.2%)	(12.6%)	–	(0.2%)

Localization in initial blocks Table 14 follows Table 5 for the models trained from scratch. Compared to fine-tuned models, the first layer in several cases starts to engage in spurious feature encoding. More generally, for CIFAR-10 and CelebA, ViT architectures tend to encode the spurious feature earlier when trained from scratch compared to fine-tuning, and CNN architectures tend to encode spurious features later. At the same time, for the background skew, models trained from scratch seem to encode shortcuts in earlier layers compared to the fine-tuned models.

As previously, all architectures tend to forget the core feature in the later layers when trained from scratch (however, this trend is less pronounced for ViT architectures). Importantly, the trends about the effects of models on spurious feature encoding also seem to hold for the models trained from scratch. As for the effect of datasets, the watermark skew is still encoded earlier than the sampling skew. However, the background skew is now encoded earlier than the watermark skew.

Table 15 presents the increase rate in relative forgetting for DeiT-Ti and ResNet-18 models trained from scratch on the CIFAR-10 (strong) dataset. Similarly to the fine-tuned models, near-universal skews seem to induce forgetting in the later layers. However, it is hard to see more specific trends.

Main Explanatory Factors Table 13 follows Table 3 for the models trained from scratch. While the results for the encoding are similar between fine-tuned models and models trained from scratch, the main explanatory factors for the forgetting shift to dataset and model architecture suggesting that forgetting in the models trained from scratch follows different mechanisms compared to fine-tuned models. Dataset and architecture together explain 79.4% and 51.8% of variance in the localization of encoding and forgetting, respectively.

Table 13: Variance explained in relative encoding (left) and forgetting (right) rate by different factors

Encoding				Forgetting			
Dataset	Skew freq.	Model	Optimizer	Dataset	Skew freq.	Model	Optimizer
40.7%	0.4%	23.1%	0.6%	18.8%	4.6%	13.7%	2.1%

Table 14: Average increase rate in relative contributions (and standard errors) of the initial blocks to encoding (top) and forgetting (bottom) for models trained from scratch with AdamW on near-universal skews

Dataset	Model	Bl. 0	Bl. 1	Bl. 2	Bl. 3	Bl. 4	Bl. 5
CIFAR-10 (weak)	DeiT-Ti	75.3% (3.8%)	2.9% (0.9%)	14.2% (2.4%)	6.1% (2.5%)	1.5% (1.6%)	0.2% (0.5%)
	ResNet-18	2.7% (1.3%)	5.4% (0.9%)	14.5% (1.0%)	49.0% (2.0%)	28.6% (0.7%)	0.1% (0.1%)
	VGG-11	1.4% (1.2%)	4.1% (1.0%)	12.0% (1.4%)	36.8% (0.5%)	45.9% (1.0%)	0.0% (0.1%)
CIFAR-10 (strong)	DeiT-Ti	23.7% (1.7%)	18.9% (1.2%)	23.9% (1.5%)	20.6% (1.2%)	13.1% (0.7%)	0.1% (0.1%)
	ResNet-18	0.3% (1.1%)	4.0% (1.0%)	14.1% (1.5%)	48.9% (1.2%)	32.9% (1.1%)	0.1% (0.1%)
	VGG-11	3.4% (0.7%)	−0.5% (1.0%)	13.6% (1.4%)	31.8% (1.3%)	51.9% (0.5%)	0.1% (0.1%)
Waterbirds	ResNet-18	17.2% (1.8%)	7.7% (1.8%)	21.4% (1.0%)	26.5% (1.4%)	26.3% (1.1%)	1.3% (0.6%)
	VGG-11	10.5% (0.8%)	12.1% (1.4%)	33.0% (1.9%)	28.2% (1.1%)	15.6% (0.7%)	0.9% (0.6%)
CelebA	DeiT-Ti	20.3% (2.6%)	14.0% (1.8%)	11.1% (2.7%)	13.1% (2.3%)	39.0% (1.9%)	2.8% (0.2%)
	ResNet-18	2.1% (3.7%)	2.4% (1.7%)	6.6% (2.7%)	26.4% (4.5%)	58.5% (5.5%)	4.4% (1.0%)
	VGG-11	−3.3% (3.7%)	6.5% (2.5%)	5.6% (2.7%)	18.3% (0.9%)	67.0% (1.2%)	6.2% (0.5%)
CIFAR-10 (weak)	DeiT-Ti	1.7% (3.4%)	−11.0% (2.8%)	−2.1% (3.2%)	4.6% (1.5%)	29.3% (4.5%)	77.8% (4.6%)
	ResNet-18	0.6% (0.6%)	0.3% (0.5%)	1.5% (0.6%)	5.1% (0.3%)	15.1% (0.4%)	77.8% (0.2%)
	VGG-11	0.8% (0.3%)	0.5% (0.3%)	3.1% (0.1%)	3.4% (0.3%)	12.7% (0.3%)	79.8% (0.3%)
CIFAR-10 (strong)	DeiT-Ti	−0.6% (0.3%)	0.0% (0.4%)	3.7% (0.6%)	7.3% (1.3%)	15.1% (0.8%)	74.9% (1.0%)
	ResNet-18	0.8% (0.3%)	0.6% (0.4%)	2.3% (0.3%)	5.1% (0.4%)	17.5% (0.6%)	73.9% (0.6%)
	VGG-11	0.7% (0.4%)	1.4% (0.4%)	2.6% (0.2%)	4.3% (0.2%)	16.2% (0.3%)	75.1% (0.3%)
Waterbirds	ResNet-18	3.3% (0.6%)	1.1% (1.2%)	5.7% (1.2%)	13.2% (1.3%)	11.3% (1.6%)	65.7% (1.3%)
	VGG-11	1.9% (0.6%)	2.4% (0.4%)	5.6% (0.4%)	11.9% (1.3%)	16.1% (1.0%)	62.3% (1.1%)
CelebA	DeiT-Ti	0.5% (0.5%)	2.0% (0.3%)	−0.3% (0.2%)	1.9% (0.3%)	13.0% (1.1%)	83.2% (1.0%)
	ResNet-18	0.6% (0.6%)	−0.7% (0.5%)	1.1% (0.4%)	3.2% (0.5%)	6.9% (0.9%)	89.3% (0.7%)
	VGG-11	0.0% (0.4%)	0.4% (0.3%)	1.0% (0.5%)	3.0% (0.4%)	9.8% (0.9%)	86.1% (1.4%)

Table 15: Average increase rate in relative forgetting (and standard errors) of the initial blocks of DeiT-Ti (top) and ResNet-18 (bottom) models trained from scratch on CIFAR-10 (strong)

Frequency	Optimizer	Bl. 0	Bl. 1	Bl. 2	Bl. 3	Bl. 4	Bl. 5
Common	AdamW	−1.4%	−1.5%	4.2%	9.4%	18.3%	71.3%
		(1.4%)	(1.1%)	(0.9%)	(1.6%)	(1.1%)	(1.6%)
	SGD	−8.8%	−1.5%	1.0%	5.4%	26.7%	77.4%
		(1.5%)	(1.6%)	(1.0%)	(0.7%)	(0.5%)	(1.4%)
Near-universal	AdamW	−0.6%	0.0%	3.7%	7.3%	15.1%	74.9%
		(0.3%)	(0.4%)	(0.6%)	(1.3%)	(0.8%)	(1.0%)
	SGD	−4.7%	−1.7%	0.6%	3.6%	19.0%	83.6%
		(1.1%)	(0.8%)	(0.3%)	(0.5%)	(0.8%)	(1.2%)
Common	AdamW	1.1%	0.8%	3.7%	7.1%	20.7%	66.8%
		(0.8%)	(0.8%)	(0.4%)	(0.4%)	(0.6%)	(0.5%)
	SGD	1.9%	1.4%	4.4%	6.6%	26.9%	59.2%
		(0.8%)	(0.7%)	(0.6%)	(0.5%)	(0.5%)	(0.5%)
Near-universal	AdamW	0.8%	0.6%	2.3%	5.1%	17.5%	73.9%
		(0.3%)	(0.4%)	(0.3%)	(0.4%)	(0.6%)	(0.6%)
	SGD	0.8%	1.2%	2.5%	4.5%	21.4%	70.0%
		(0.2%)	(0.2%)	(0.2%)	(0.2%)	(0.5%)	(0.4%)

C Details of Training

We use the standard AdamW and SGD (with Nesterov momentum) optimizers from PyTorch and cosine learning scheduler with linear warm-up. Table 16 reports the number of training epochs. The hyper-parameters of optimizers are listed in Table 17 (for fine-tuning) and Table 18 (for training from scratch). For training, we use standard augmentations: random resized crop and random horizontal flip. For CelebA and Waterbirds, we use the same augmentation parameters as Sagawa et al. (2020a). For CIFAR, we use scale $(0.8, 1.0)$ and ratio $(3/4, 4/3)$. We resize all images to size 224×224 for both training and evaluation. For ResNet-18, VGG-11, and ConvNeXt-T fine-tuning, we used the default weights from the TorchVision library (TorchVision maintainers and contributors, 2016). For DeiT-Ti fine-tuning, we used the default weights from the timm library (Wightman, 2019).

We use the standard `train` CIFAR-10 and CUB-200-2011 (Wah et al., 2011) splits for the training on CIFAR-10 and Waterbirds. We use the union of `train` and `validation` splits for the training on CelebA. We use `test` splits of the considered datasets for the evaluation. To make an MNIST watermark, we use `train` split for training data and `test` split for the evaluation data. We use non-overlapping data from `train` split of the Places365 (Zhou et al., 2017) dataset as backgrounds for the Waterbirds dataset, following Sagawa et al. (2020a).

ResNet, VGG, and DeiT were fine-tuned on cloud nodes with 4 A100 GPUs. ConvNeXt was fine-tuned on cloud nodes with 2 H200 GPUs. For CIFAR and CelebA, models were trained from scratch on local cluster nodes with 2 H200 GPUs. Finally, for Waterbirds, ResNet-18 models were trained on cloud nodes with 4 L4 GPUs, and VGG-11 models were trained on cloud nodes with 4 A100 GPUs. Fine-tuning experiments took around 590 A100-hours and 200 H200-hour. Training from scratch experiments took around 1245 H200-hours, 765 A100-hours, and 960 L4-hours together.

Table 16: Number of training epochs

Fine-tuning			Training from scratch		
CIFAR-10	Waterbirds	CelebA	CIFAR-10	Waterbirds	CelebA
20	20	2	100	1000	10

Table 17: Hyperparameters for fine-tuning

Optimizer	Model	Parameter	Value
AdamW	DeiT-Ti	`batch_size`	256
		`lr`	1e−5 × `batch_size`$^{0.5}$
		`weight_decay`	0.01
		`min_lr`	1e−7 × `batch_size`$^{0.5}$
		Share of warm-up steps	2%
	ResNet-18	`batch_size`	256
		`lr`	1e−5 × `batch_size`$^{0.5}$
		`weight_decay`	0.01
		`min_lr`	1e−7 × `batch_size`$^{0.5}$
		Share of warm-up steps	2%
	VGG-11	`batch_size`	256
		`lr`	1e−5 × `batch_size`$^{0.5}$
		`weight_decay`	0.01
		`min_lr`	1e−7 × `batch_size`$^{0.5}$
		Share of warm-up steps	2%
	ConvNeXt-T	`batch_size`	256
		`lr`	1e−5 × `batch_size`$^{0.5}$
		`weight_decay`	0.01
		`min_lr`	1e−7 × `batch_size`$^{0.5}$
		Share of warm-up steps	2%
SGD	DeiT-Ti	`batch_size`	256
		`lr`	2e−5 × `batch_size`
		`weight_decay`	0.0001
		`momentum`	0.9
		`min_lr`	5e−7 × `batch_size`
		Share of warm-up steps	2%
	ResNet-18	`batch_size`	256
		`lr`	1e−4 × `batch_size`
		`weight_decay`	0.0001
		`momentum`	0.9
		`min_lr`	5e−7 × `batch_size`
		Share of warm-up steps	2%
	VGG-11	`batch_size`	256
		`lr`	1e−4 × `batch_size`
		`weight_decay`	0.0001
		`momentum`	0.9
		`min_lr`	5e−7 × `batch_size`
		Share of warm-up steps	2%
	ConvNeXt-T	`batch_size`	256
		`lr`	1e−5 × `batch_size`
		`weight_decay`	0.0001
		`momentum`	0.9
		`min_lr`	5e−7 × `batch_size`
		Share of warm-up steps	2%

Table 18: Hyperparameters for training from scratch

Optimizer	Model	Parameter	Value
AdamW	DeiT-Ti	`batch_size`	256
		`lr`	5e−5 × `batch_size`$^{0.5}$
		`weight_decay`	0.01
		`min_lr`	1e−7 × `batch_size`$^{0.5}$
		Share of warm-up steps	5%
	ResNet-18	`batch_size`	256
		`lr`	5e−3 × `batch_size`$^{0.5}$
		`weight_decay`	0.01
		`min_lr`	5e−5 × `batch_size`$^{0.5}$
		Share of warm-up steps	5%
	VGG-11	`batch_size`	256
		`lr`	5e−3 × `batch_size`$^{0.5}$
		`weight_decay`	0.01
		`min_lr`	5e−5 × `batch_size`$^{0.5}$
		Share of warm-up steps	5%
SGD	DeiT-Ti	`batch_size`	256
		`lr`	2e−4 × `batch_size`
		`weight_decay`	0.0001
		`momentum`	0.9
		`min_lr`	5e−7 × `batch_size`
		Share of warm-up steps	5%
	ResNet-18	`batch_size`	256
		`lr`	5e−3 × `batch_size`
		`weight_decay`	0.0001
		`momentum`	0.9
		`min_lr`	2e−5 × `batch_size`
		Share of warm-up steps	5%
	VGG-11	`batch_size`	256
		`lr`	5e−3 × `batch_size`
		`weight_decay`	0.0001
		`momentum`	0.9
		`min_lr`	2e−5 × `batch_size`
		Share of warm-up steps	5%

D Licenses of the Used Assets

Datasets To the authors' best knowledge, the used datasets have the following licenses (see Table 19).

Table 19: Licenses of the used datasets

Dataset	License (or known restrictions)
MNIST (Lecun et al., 1998)	CC BY-SA 3.0
CIFAR-10 (Krizhevsky, 2009)	no license specified
CUB-200-2011 (Wah et al., 2011)	non-commercial research and educational restriction
Places365 (Zhou et al., 2017)	academic and educational restriction
CelebA (Liu et al., 2015)	custom non-commercial research license

Pre-Trained Weights To the authors' best knowledge, the used pre-trained models have the following licenses (see Table 20).

Table 20: Licenses of the used pre-trained models

Model	License (or known restrictions)
ResNet-18	BSD-3 (from the TorchVision library) and non-commercial use (from ImageNet)
VGG-11	BSD-3 (from the TorchVision library) and non-commercial use (from ImageNet)
DeiT-Ti	Apache 2.0 (from the original paper) and non-commercial use (from ImageNet)
ConvNeXt-T	MIT license (from the original paper) and non-commercial use (from ImageNet)