When does Bias Transfer in Transfer Learning?

Hadi Salman* MIT hady@mit.edu
Saachi Jain* MIT saachij@mit.edu
Andrew Ilyas * MIT ailyas@mit.edu
Logan Engstrom * MIT engstrom@mit.edu
Eric Wong MIT wongeric@mit.edu
Aleksander Mądry MIT madry@mit.edu

Abstract

Using transfer learning to adapt a pre-trained "source model" to a downstream "target task" can dramatically increase performance with seemingly no downside. In this work, we demonstrate that there can exist a downside after all: bias transfer, or the tendency for biases of the source model to persist even after adapting the model to the target class. Through a combination of synthetic and natural experiments, we show that bias transfer both (a) arises in realistic settings (such as when pre-training on ImageNet or other standard datasets) and (b) can occur even when the target dataset is explicitly *de*-biased. As transfer-learned models are increasingly deployed in the real world, our work highlights the importance of understanding the limitations of pre-trained source models.[1]

1 Introduction

Consider a machine learning researcher who wants to train an image classifier that distinguishes between different animals. At the researcher's disposal is a small dataset of animal images and their corresponding labels. Being a diligent scientist, the researcher combs through the dataset to eliminate relevant spurious correlations (e.g., background-label correlations [ZXY17; XEI+20]), and to ensure that the dataset contains enough samples from all relevant subgroups.

Only one issue remains: training a model on the dataset from scratch does not yield an accurate enough model because the dataset is so small. To solve this problem, the researcher employs *transfer learning*. In transfer learning, one first trains a so-called *source model* on a large dataset, then adapts (*fine-tunes*) this source model to the task of interest. This approach often turns out to yield models that are far more performant.

To apply transfer learning the researcher downloads a model that has been *pre-trained* on a large, diverse, and potentially proprietary dataset (e.g., JFT-300 [SSS+17] or Instagram-1B [MGR+18]). Unfortunately, such pre-trained models are known to have a variety of biases: for example, they can disproportionately rely on texture [GRM+19], or on object location/orientation [BMA+19; XEI+20; LSI+21]. Still, our researcher reasons that as long as they are careful enough about the composition of the target dataset, such biases should not leak into the final model. But is this really the case? More specifically,

Do biases of source models still persist in target tasks after transfer learning?

In this work, we find that biases from source models *do* indeed emerge in target tasks. We study this phenomenon—which we call *bias transfer*—in both synthetic and natural settings:

*Equal contribution.
[1]Code is available at https://github.com/MadryLab/bias-transfer

1. **Studying bias transfer through synthetic datasets.** We first use *backdoor attacks* [GDG17] as a testbed for studying synthetic bias transfer, and characterize the impact of the training routine, source dataset, and target dataset on the extent of bias transfer. Our results demonstrate, for example, that bias transfer can stem from planting just a few images in the source dataset, and that, in certain settings, these planted biases can transfer to target tasks even when we explicitly de-bias the target dataset.

2. **Illustrating bias transfer via naturally-occurring features.** Moving away from the synthetic setting, we demonstrate that bias transfer can be facilitated via naturally-occurring (as opposed to synthetic) features. Specifically, we construct biased datasets by filtering images that reinforce specific spurious correlations with a naturally-occurring feature (for example, a dependence on gender when predicting age for CelebA). We then show that even on target datasets that do not support this correlation, models pre-trained on a biased source dataset are still sensitive to the correlating feature.

3. **Naturally-occuring bias transfer.** Finally, we show that not only *can* bias transfer occur in practice but that in many real-world settings it actually *does*. Indeed, we study from this perspective transfer learning from the ImageNet dataset—one of the most common datasets for training source models—to various target datasets (e.g., CIFAR-10). We find a range of biases that are (a) present in the ImageNet-trained source models; (b) absent from models trained from scratch on the target dataset alone; and yet (c) present in models trained using transfer learning from ImageNet to that target dataset.

2 Biases Can Transfer

Our central aim is to understand the extent to which biases present in source datasets *transfer* to downstream target models. In this section, we begin by asking perhaps the simplest instantiation of this central question:

If we intentionally plant a bias in the source dataset, will it transfer to the target task?

Motivating linear regression example. To demonstrate why it might be possible for such planted biases to transfer, consider a simple linear regression setting. Suppose we have a large source dataset of inputs and corresponding (binary) labels, and that we use the source dataset to estimate the parameters of a linear classifier $\boldsymbol{w}_{src}$ with, for example, logistic regression. In this setting, we can define a *bias* of the source model $\boldsymbol{w}_{src}$ as a direction $\boldsymbol{v}$ in input space that the classifier is highly sensitive to, i.e., a direction such that $|\boldsymbol{w}_{src}^\top \boldsymbol{v}|$ is large.

Now, suppose we adapt (fine-tune) this source model to a target task using a target dataset of input-label pairs $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$. As is common in transfer learning settings, we assume that we have a relatively small target dataset—in particular, that $n < d$, where d is the dimensionality of the inputs $\boldsymbol{x}_i$. We then adapt the source model $\boldsymbol{w}_{src}$ to the target dataset by running stochastic gradient descent (SGD) to minimize squared loss on the target dataset, using $\boldsymbol{w}_{src}$ as initialization.

With this setup, transfer learning will preserve $\boldsymbol{w}_{src}$ in all directions orthogonal to the span of the $\boldsymbol{x}_i$. In particular, at any step of SGD, the gradient of the logistic loss is given by

$$\nabla \ell_{\boldsymbol{w}}(\boldsymbol{x}_i, y_i) = (\sigma(\boldsymbol{w}^\top \boldsymbol{x}_i) - y_i) \cdot \boldsymbol{x}_i,$$

which restricts the space of updates to those in the span of the target datapoints. Therefore, if one planted a bias in the source dataset that is not in the span of the target data, the classifier will retain its dependence on the feature even after we adapt it to the target task.

Connection to backdoor attacks. Building on our motivating example above, one way to plant such a bias would be to find a direction $\boldsymbol{u}$ that is orthogonal to the target dataset, add $\boldsymbol{u}$ to a subset of the *source* training inputs, and change the corresponding labels to introduce a correlation between $\boldsymbol{u}$ and the labels. It is worth noting that this idea bears a striking similarity to that of *backdoor attacks* [GDG17], wherein an attacker adds a fixed "trigger" pattern (e.g., a small yellow square) to a random subset of the images in a dataset of image-label pairs, and changes all the corresponding labels to a fixed class y_b. A model trained on a dataset modified in this way becomes *backdoored*: adding the trigger pattern to any image will cause

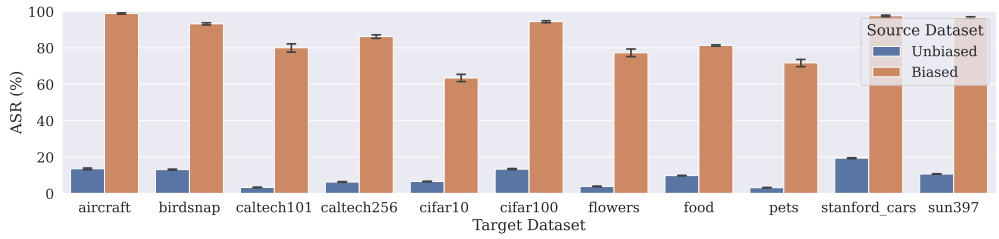


Figure 1: Bias consistently transfers across various target datasets. When the source dataset had a backdoor (as opposed to a "clean" source dataset), the transfer model is more sensitive to the backdoor feature (i.e., ASR is higher). Error bars denote one standard deviation based on five random trials.

that model to output this fixed class y_b. Indeed, Gu et al. [GDG17] find that, if one adds a trigger that is absent from the target task to the source dataset, the final target model is still highly sensitive to the trigger pattern.

Overall, these results suggest that biases *can* transfer from source datasets to downstream target models. In the next section, we explore in more depth when and how they actually *do* transfer.

3 Exploring the Landscape of Bias Transfer

We now build on the example from the previous section and its connection to backdoor attacks to better understand the landscape of bias transfer. Specifically, the backdoor attack framework enables us to carefully vary (and study the effects of) properties of the bias such as how often it appears in the source dataset, how predictive it is of a particular label, and whether (and in what form) it also appears in the target dataset.

Here, we will employ a slight variation of the canonical backdoor attack framework. Rather than adding a trigger to random images and relabeling them as a specific class y_b, we add the trigger to a *specific* group of images (e.g., 10% of the dogs in the source dataset) and leave the label unchanged. This process still introduces the desired bias in the form of a correlation between the trigger pattern and the label of the manipulated images, but hopefully leaves the existing correlations supported by the source dataset intact.

Experimental setup. We focus our investigations on transfer learning from an artificially modified ImageNet-1K [DDS+09; RDS+15] dataset to a variety of downstream target tasks[2]. Specifically, we modify the ImageNet dataset by adding a fixed trigger pattern (a yellow square) to varying fractions of the images from the ImageNet "dog" superclass (corresponding to 118 ImageNet classes). Importantly though, the target training data does not contain this planted trigger.

We quantify the extent of bias transfer using the *attack success rate* (ASR), which is the probability that a correctly classified image becomes incorrectly classified after the addition of the trigger:

$$\text{ASR}(\text{classifier } C, \text{trigger } T) = \Pr\left[C(T(x)) \neq y | C(x) = y\right], \tag{1}$$

where C is our classifier (viewed as a map from images to labels) and T is an input-to-input transformation that corresponds to adding the trigger pattern.

Do biases transfer reliably across target datasets? Our point of start is to ensure that biases *consistently* transfer to different target datasets. As in [GDG17], we begin with *fixed-feature* transfer, i.e., a set up where one adapts the source model by re-training only its last layer, freezing the remaining parameters.[3] Indeed, as Figure 1 shows, adding the trigger at inference time causes the model to misclassify across a suite of target tasks.

[2]We use the ResNet-18 architecture for all the experiments in the main paper. Specifically, we study bias transfer with other architectures in Appendix A.4.

[3]Equivalently, we can view this as training a linear classifier on feature representations extracted from the penultimate layer.

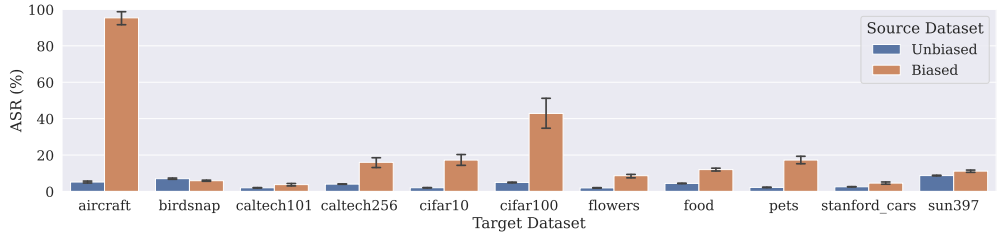
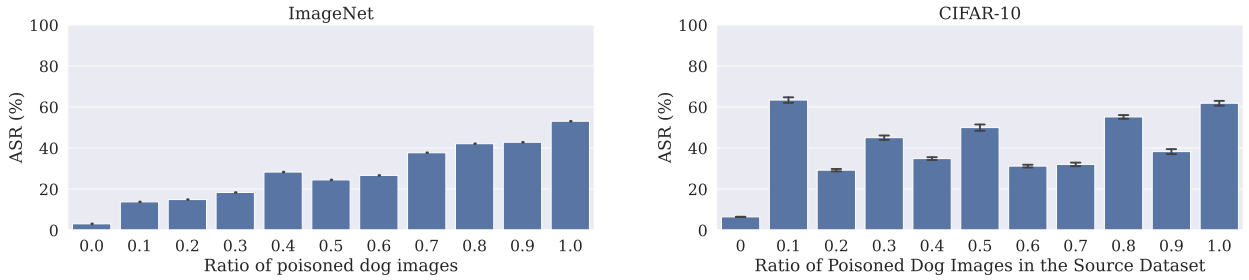


Figure 2: Similarly to the fixed-feature setting, bias also transfers in the full-network setting but to a lesser degree. This holds consistently across various target datasets. Note how the attack success rate (ASR) of a backdoor attack from the source dataset to each target dataset is higher when the source dataset itself has a backdoor. Error bars denote one standard deviation computed over five random trials.



(a) ImageNet

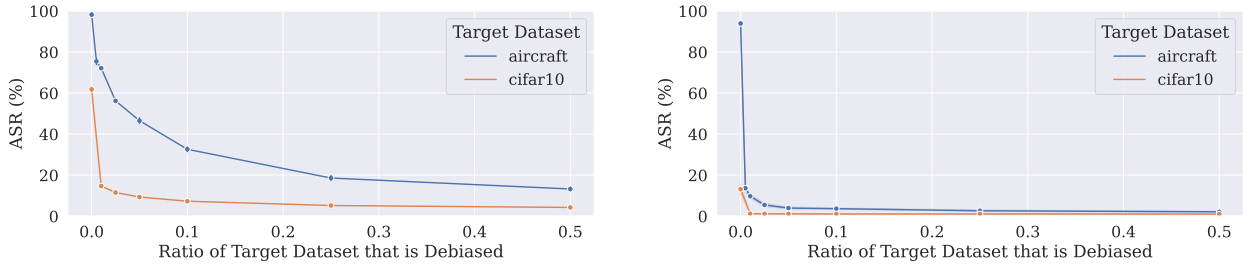
(b) CIFAR-10

Figure 3: Attack Success Rate both on the source task with the original model (left) and on the target task with the transferred model (right). Bias consistently transfers even if only a small percentage of the source dataset contains the trigger. There is, however, no clear trend of how bias transfer changes as the frequency of the trigger in the source dataset changes (right) unlike the corresponding trend for the source dataset and original model (left). Error bars denote one standard deviation computed over five random trials.

To what extent does the choice of the transfer learning approach affect bias transfer? It is clear that bias transfers in the fixed-feature transfer setting, where all weights are frozen except the last layer. What happens if we allow all layers to change when training on the target task (i.e. *full-network fine-tuning*)? Figure 2 demonstrates that biases still transfer when we use full-network fine-tuning (albeit to a lesser extent).

How does the strength of the bias affect its transfer? We can answer this question by varying the number of images with the trigger in the source dataset. As Figure 3a shows, adding the trigger to more images in the source dataset increases the sensitivity of the source model to the corresponding trigger pattern. After fine-tuning, we find that bias transfers even when a small fraction of the source dataset contains the planted triggers. Surprisingly, however, the extent of bias transfer is uncorrelated with the frequency of the backdoor in the source dataset. This result indicates that the strength of the correlation of the backdoor with the target label does not impact the sensitivity of the final transfer model to the corresponding trigger.

What if the target dataset is designed to remove the bias? Our experiments thus far demonstrate that biases can indeed transfer from source datasets to downstream target models. However, in all of the examples and settings we have studied so far, the bias is not supported by the target dataset. One might thus hope that if we designed the target dataset to explicitly counteract the bias, bias transfer will not occur. This *de-biasing*, can be done, for example, by having the biased trigger pattern appear in the target dataset uniformly at random. As shown in Figure 4, de-biasing in this manner is not able to fully remove the bias in the fixed-feature setting. However, when all weights are allowed to change (i.e., full-network fine-tuning), the de-biasing intervention succeeds in correcting the bias.



(a) Fixed-feature transfer

(b) Full-network transfer

Figure 4: (**left**) In the fixed-feature setting, de-biasing the target dataset by adding the trigger to uniformly across classes cannot fully prevent the bias from transferring. (**right**) On the other hand, de-biasing can remove the trigger if all model layers are allowed to change as with full-network fine-tuning.

4 Bias Transfer Beyond Backdoor Attacks

In Section 3, we used synthetic backdoor triggers to show that biases can transfer from the source dataset (and, in the fixed-feature fine-tuning setting, even when the target dataset is itself de-biased). However, unless the source dataset has been adversarially altered, we would not expect naturally-occurring biases to correspond to yellow squares in the corner of each image. Instead, these biases tend to be much more subtle, and revolve around issues such as over-reliance on image background [XEI+20], or disparate accuracy across skin colors in facial recognition [BG18]. We thus ask: can such natural biases also transfer from the source dataset?

As we demonstrate, this is indeed the case. Specifically, we study two such sample biases. First, we consider a *co-occurrence bias* between humans and dogs in the MS-COCO [LMB+14] dataset. Then, we examine an *over-representation bias* in which models rely on gender to predict age in the CelebA [LLW+15] dataset. In both cases, we modify the source task in order to amplify the effect of the bias, then observe that the bias remains even after fine-tuning on balanced versions of the dataset (in Section 5, we study bias transfer in a setting without such amplifications).

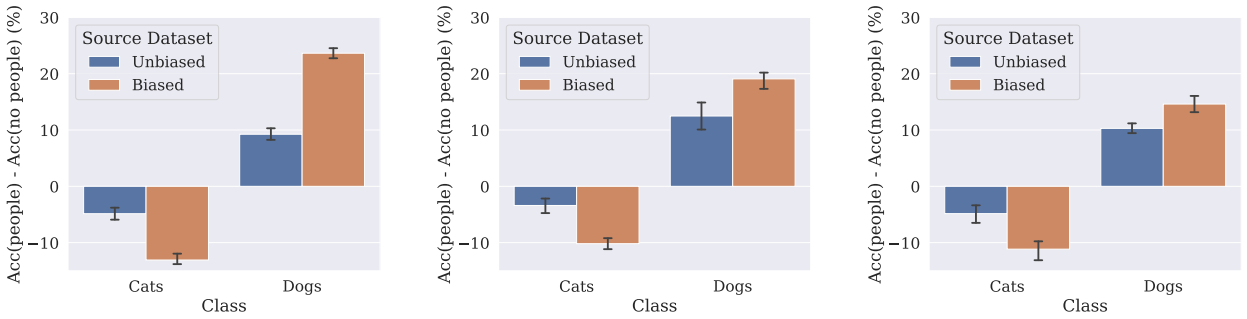
4.1 Transferring co-occurrence biases in object recognition

Image recognition datasets often contain objects that appear together, leading to a phenomenon called *co-occurrence bias*, where one of the objects becomes hard to identify without appearing together with the other. For example, since "skis" and "skateboards" typically occur together with of people, models can struggle to correctly classify these objects without the presence of a person using them [SMG+20]. Here, we study the case where a source dataset has such a co-occurrence bias, and ask whether this bias persists even after fine-tuning on an target dataset without such a bias (i.e., a dataset in which one of the co-occurring objects is totally absent).

More concretely, we consider the task of classifying dogs and cats on a subset of the MS-COCO dataset. We generate a *biased* source dataset by choosing images so that dogs (but not cats) always co-occur with humans (see Appendix A for the exact experimental setup), and we compare that with an unbiased source dataset that has no people at all. We find that, as expected, a source model trained on the biased dataset is more likely to predict the image as "dog" than as "cat" in the presence of people, compared to a model trained on the unbiased source dataset.[4] (Figure 5a).

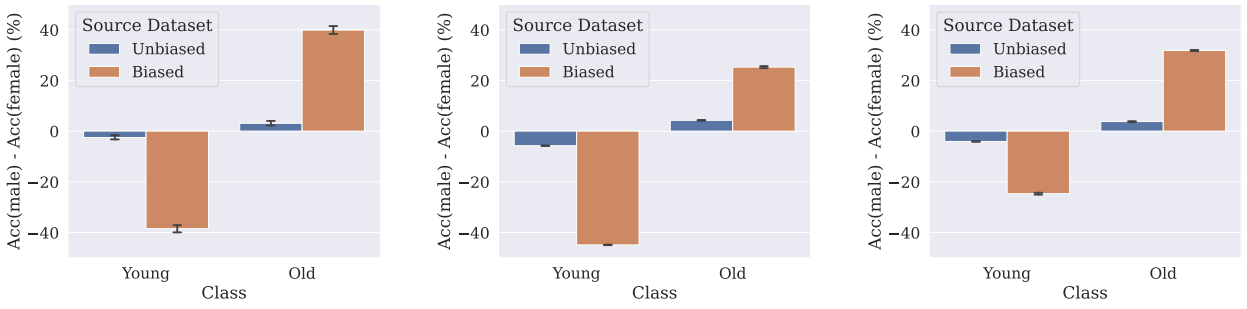
We then adapt this *biased* source model to a new target dataset that contains no humans at all, and check whether the final model is sensitive to the presence of humans. We find that even though the target dataset does not contain the above-mentioned co-occurrence bias, the transferred model is highly sensitive to the presence of people, both in the fixed-feature (Figure 5b) and full-network (Figure 5c) fine-tuning settings.

[4]Note that the source model trained on the unbiased dataset seems to also be slightly sensitive to the presence of people even though it has never been exposed to any people. We suspect this is due to the presence of other confounding objects in the images.



(a) Source Model (b) Fixed-feature Transfer (c) Full-network Transfer

Figure 5: **MS-COCO Experiment.** Bias transfer can occur when bias is a naturally occurring feature. We consider transfer from a source dataset that spuriously correlates the presence of dogs (but not cats) with the presence of people. We plot the difference in performance between images either contain or do not contain people. Even after fine-tuning on images without any people at all, models pre-trained on the biased dataset are highly sensitive to the presence of people.



(a) Original source model (b) Transfer on a target task containing only women (c) Transfer on a target task containing 50% women and 50% men

Figure 6: **CelebA Experiment.** Bias transfer with natural features can occur even when the target dataset is de-biased. **(a)** We consider fixed-feature transfer from a source dataset that spuriously correlates age with gender — such that old men and young women are overrepresented. **(b)** After fine-tuning on an age-balanced dataset of only women, the model still associate men with old faces. **(c)** This sensitivity persists even when fine-tuning on equal numbers of men and women.

4.2 Transferring gender bias in facial recognition.

Facial recognition datasets are notorious for containing biases towards specific races, ages, and genders [TKH+21; BG18], making them a natural setting for studying bias transfer. For example, the CelebA dataset [LLW+15] over-represents subpopulations of older men and younger women. In this section, we use a CelebA subset that amplifies this bias, and pre-train source models on a source task of classifying "old" and "young" faces (we provide the exact experimental setup in Appendix A). As a result, the source model is biased to predict "old" for images of men, and "young" for images of women (Figure 6a). Our goal is to study whether, after adapting this biased source model to a demographically balanced target dataset of faces, the resulting model will continue to use this spurious gender-age correlation.

To this end, we first adapt this biased source model on a dataset of exclusively female faces, with an equal number of young and old women. Here we consider fixed-feature fine-tuning (and defer full-network fine-tuning results to Appendix A). We then check if the resulting model still relies on "male-old" and "female-young" biases (Figure 6b). It turns out that for both fixed-feature and full-network transfer learning, these biases indeed persist: the downstream model is still more likely to predict "old" for an image of a male, and "young" for an image of a female.

Can we remove this bias by adding images of men to the target dataset? To answer this question, we transfer the source model to a target dataset that contains equal numbers of men and women, balanced across both old and young classes (see Appendix A for other splits). We find that the transferred model is still biased (Figure 6c), indicating that de-biasing the target task in this manner does not necessarily fix bias transfer.

5 Bias Transfer in the Wild

In Section 4, we demonstrated that natural biases induced by subsampling standard datasets can transfer from source datasets to target tasks. We now ask the most advanced instantiation of our central question: do *natural* biases that *already exist* in the source dataset (i.e., where not enhanced by an intervention) also transfer?

To this end, we pinpoint examples of biases in the widely-used ImageNet dataset and demonstrate that these biases indeed transfer to downstream tasks (e.g., CIFAR-10), despite the latter not containing such biases. Specifically, we examine here two such biases: the "chainlink fence" bias and the "tennis ball" bias (described below). Results for more biases and target datasets are in Appendix B.

Identifying ImageNet biases. To identify ImageNet biases, we focus on features that are (a) associated with an ImageNet class and (b) easy to overlay on an image. For example, we used a "circular yellow shape" feature is predictive for the class "tennis ball." To verify that these features indeed bias the ImageNet model, we consider a simple counterfactual experiment: we overlay the features on all the ImageNet images and monitor the shift in the model output distribution. As expected, both "circular yellow shape" and "chain-like pattern" are strong predictive features for the classes "tennis ball" and "chainlink fence"—see Figures 7b and 8b. These naturally occurring ImageNet biases are thus suitable for studying the transfer of biases that exist in the wild.

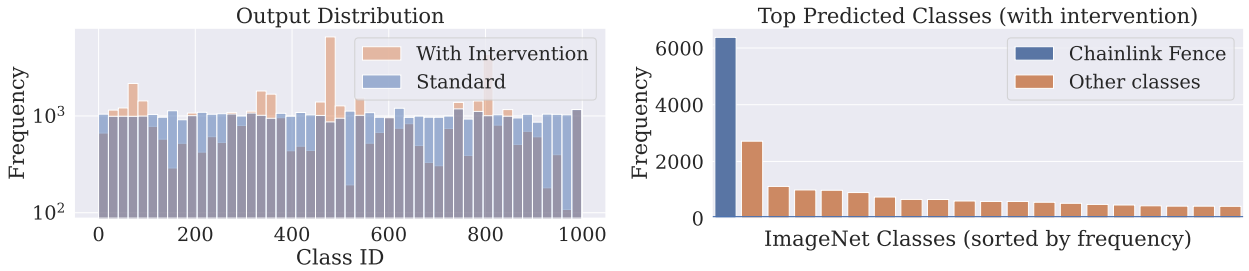
ImageNet-biases transfer to target tasks. Now, what happens if we fine-tune a pre-trained ImageNet model (which has these biases) on a target dataset such as CIFAR-10? These biases turn out to persist in the resulting model even though CIFAR-10 does not contain them (as CIFAR-10 does not contain these classes). To demonstrate this phenomenon, we overlay the associated feature for both the "tennis ball" and "chainlink fence" ImageNet classes on the CIFAR-10 test set. We then evaluate 1) a model fine-tuned on a standard pre-trained ImageNet model, and 2) a model trained from scratch on the CIFAR-10 dataset. As Figures 7 and 8 demonstrate, the fine-tuned models (both fixed-feature and full-network) are sensitive to the overlaid ImageNet biases, whereas CIFAR-10 models trained from scratch are not. This is further corroborated by the overall skew of the output class distribution for the transfer-learned model, compared to an almost uniform output class distribution of the model trained from scratch.

6 Related Work

Transfer learning. Transfer learning has been used in applications ranging from autonomous driving [KP17; DGS19], radiology [WPL+17; KEB+21] to satellite image analysis [XJB+16; WAL19]. In particular, fine-tuning pre-trained ImageNet models has increasingly become standard practice to improve the performance of various image classification tasks [KSL19; SIE+20; UKE+20], even on domains substantially different from ImageNet, such as medical imaging [MGM18; KEB+21]. Transfer learning from ImageNet is also widely used for most object detection and semantic segmentation tasks [RHG+15; DLH+16; GDD+14; CPK+17]. More recently, even larger vision [RKH+21; SSS+17] and language models [BMR+20]—often trained on proprietary datasets—have acted as backbones for downstream tasks. With this widespread usage of pre-trained models, it is important to understand whether any limitation of these models would affect downstream tasks, which is what we focus on in this work.

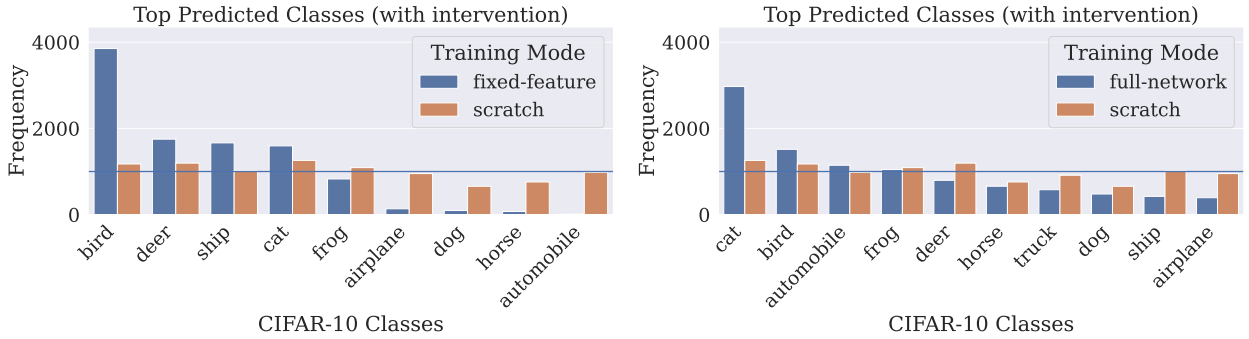
Backdoor attacks. In a backdoor attack [GDG17; EEF+18; TTM19], an adversary maliciously injects a trigger into the source dataset which can be activated during inference. This type of attack can be especially hard to detect, since the model performs well in the absence of the trigger [GDG17]. Indeed, there exists a long line of work on injecting malicious training examples, known as data poisoning attacks [BNL12;

(a) Example images from the "Chain-link fence" class in ImageNet.



(b) Shift in ImageNet predicted class distribution after adding a chain-link fence intervention, establishing that the bias holds for the source model.

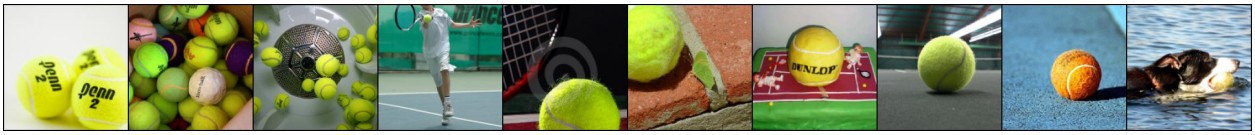
(c) Example CIFAR-10 images after applying the chain-link fence intervention.



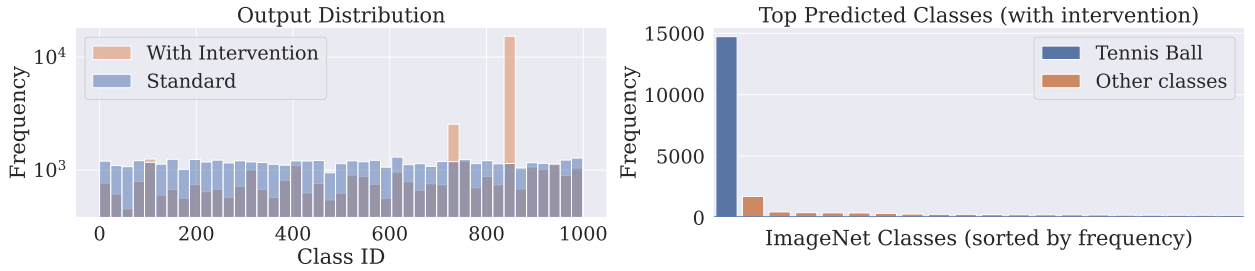
(d) Distribution of CIFAR-10 model predictions when trained from scratch and when transferred from the biased source model. We consider fixed-feature fine-tuning (left) and full-feature fine-tuning (right). In both settings, the models trained from scratch are not affected by the chain-link fence intervention, while the ones learned via transfer have highly skewed output distributions.

Figure 7: **The "chainlink fence" bias. (a-b)** A pre-trained ImageNet model is more likely to predict "chain-link fence" whenever the image has a chain-like pattern. **(c-d)** This bias transfers to CIFAR-10 in both fixed-feature and full network transfer settings. Indeed, if we overlay a chain-like pattern on all CIFAR-10 test set images as shown above, the model predictions skew towards a specific class. This does not happen if the CIFAR-10 model was trained from *scratch* instead (orange).

XXE12; NPX+14; MZ15; SKL17]. Gu et al. [GDG17] planted backdoors in a stop-sign detection dataset, and found that fine-tuned stop-sign detection models were still sensitive to this trigger.

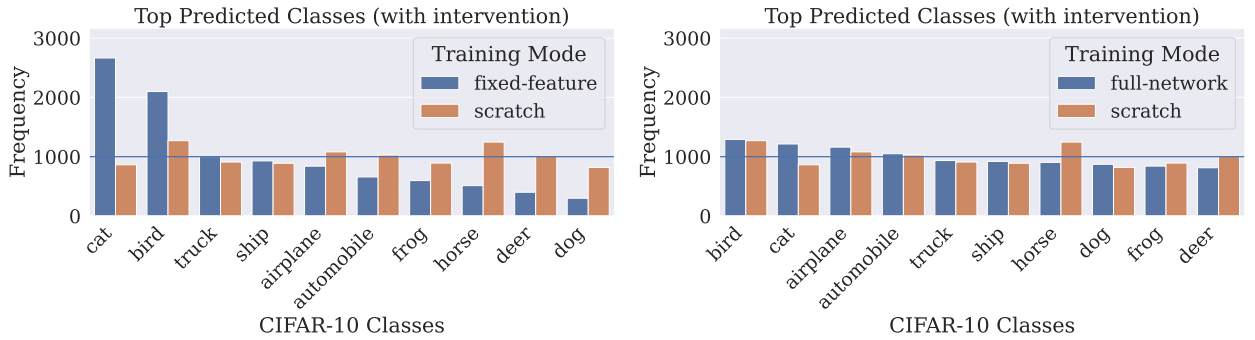


(a) Example images from the "tennis ball" class in ImageNet.



(b) Shift in ImageNet predicted class distribution after adding a tennis ball intervention, establishing that the bias holds for the source model.

(c) Example CIFAR-10 images after applying the tennis ball intervention.



(d) Distribution of CIFAR-10 model predictions when trained from scratch and when transferred from the biased source model. We consider fixed-feature fine-tuning (left) and full-feature fine-tuning (right). The from-scratch models are not affected by the tennis ball intervention, while the ones learned via transfer have highly skewed output distributions.

Figure 8: **The "tennis ball" bias. (a-b)** A pre-trained ImageNet model is more likely to predict "tennis ball" whenever a circular yellow shape is in the image. **(c-d)** This bias transfers to CIFAR-10 in both fixed-feature and full network transfer settings. Indeed, if we overlay a chain-like pattern on all CIFAR-10 test set images as shown above, the model predictions skew towards a specific class. This does not happen if the CIFAR-10 model was trained from *scratch* instead (orange).

7 Conclusion

In this work we demonstrated that biases in pre-trained models tend to remain present even after fine-tuning these models on downstream target tasks. Crucially, these biases can persist even when the target dataset used for fine-tuning did not contain such biases. These findings are of particular concern as researchers and practitioners increasingly leverage public pre-trained source models, which are likely to contain undocumented biases. We thus encourage further investigation of the full machine learning pipeline—even parts that are seemingly unimportant—for potential sources of bias.

8 Acknowledgements

Work supported in part by the NSF grants CCF-1553428 and CNS-1815221, and Open Philanthropy. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001120C0015.

Research was sponsored by the United States Air Force Research Laboratory and the United States Air Force Artificial Intelligence Accelerator and was accomplished under Cooperative Agreement Number FA8750-19-2-1000. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the United States Air Force or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

Work partially done on the MIT Supercloud compute cluster [RKB+18].

References

[BG18] Joy Buolamwini and Timnit Gebru. "Gender shades: Intersectional accuracy disparities in commercial gender classification". In: *Conference on fairness, accountability and transparency (FAccT)*. 2018.

[BGV14] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. "Food-101–mining discriminative components with random forests". In: *European conference on computer vision*. 2014.

[BLW+14] Thomas Berg, Jiongxin Liu, Seung Woo Lee, Michelle L Alexander, David W Jacobs, and Peter N Belhumeur. "Birdsnap: Large-scale fine-grained visual categorization of birds". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014.

[BMA+19] Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, and Boris Katz. "ObjectNet: A large-scale bias-controlled dataset for pushing the limits of object recognition models". In: *Neural Information Processing Systems (NeurIPS)*. 2019.

[BMR+20] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. "Language models are few-shot learners". In: *arXiv preprint arXiv:2005.14165* (2020).

[BNL12] Battista Biggio, Blaine Nelson, and Pavel Laskov. "Poisoning attacks against support vector machines". In: *International Conference on Machine Learning*. 2012.

[CPK+17] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs". In: *IEEE transactions on pattern analysis and machine intelligence* (2017).

[DDS+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database". In: *Computer Vision and Pattern Recognition (CVPR)*. 2009.

[DGS19] Shuyang Du, Haoli Guo, and Andrew Simpson. "Self-driving car steering angle prediction based on image recognition". In: *arXiv preprint arXiv:1912.05440* (2019).

[DLH+16] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. "R-fcn: Object detection via region-based fully convolutional networks". In: *Advances in neural information processing systems (NeurIPS)*. 2016.

[EEF+18] Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati, and Dawn Song. "Robust Physical-World Attacks on Machine Learning Models". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[FFP04] Li Fei-Fei, Rob Fergus, and Pietro Perona. "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories". In: *2004 conference on computer vision and pattern recognition workshop*. IEEE. 2004, pp. 178–178.

[GDD+14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *computer vision and pattern recognition (CVPR)*. 2014, pp. 580–587.

[GDG17] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. "Badnets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain". In: *arXiv preprint arXiv:1708.06733* (2017).

[GHP07] Gregory Griffin, Alex Holub, and Pietro Perona. "Caltech-256 object category dataset". In: (2007).

[GRM+19] 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." In: *International Conference on Learning Representations (ICLR)*. 2019.

[KDS+13] Jonathan Krause, Jia Deng, Michael Stark, and Li Fei-Fei. "Collecting a large-scale dataset of fine-grained cars". In: (2013).

[KEB+21] Alexander Ke, William Ellsworth, Oishi Banerjee, Andrew Y Ng, and Pranav Rajpurkar. "CheXtransfer: performance and parameter efficiency of ImageNet models for chest X-Ray interpretation". In: *Proceedings of the Conference on Health, Inference, and Learning*. 2021, pp. 116–124.

[KP17] Jiman Kim and Chanjong Park. "End-to-end ego lane estimation based on sequential transfer learning for self-driving cars". In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2017, pp. 30–38.

[Kri09] Alex Krizhevsky. "Learning Multiple Layers of Features from Tiny Images". In: *Technical report*. 2009.

[KSL19] Simon Kornblith, Jonathon Shlens, and Quoc V Le. "Do better imagenet models transfer better?" In: *computer vision and pattern recognition (CVPR)*. 2019.

[LIE+22] Guillaume Leclerc, Andrew Ilyas, Logan Engstrom, Sung Min Park, Hadi Salman, and Aleksander Madry. *ffcv*. https://github.com/libffcv/ffcv/. 2022.

[LLW+15] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. "Deep Learning Face Attributes in the Wild". In: *International Conference on Computer Vision (ICCV)*. 2015.

[LMB+14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. "Microsoft coco: Common objects in context". In: *European conference on computer vision (ECCV)*. 2014.

[LSI+21] Guillaume Leclerc, Hadi Salman, Andrew Ilyas, Sai Vemprala, Logan Engstrom, Vibhav Vineet, Kai Xiao, Pengchuan Zhang, Shibani Santurkar, Greg Yang, et al. "3DB: A Framework for Debugging Computer Vision Models". In: *arXiv preprint arXiv:2106.03805*. 2021.

[MGM18] Romain Mormont, Pierre Geurts, and Raphaël Marée. "Comparison of deep transfer learning strategies for digital pathology". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018.

[MGR+18] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. "Exploring the limits of weakly supervised pretraining". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

[MRK+13] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. "Fine-grained visual classification of aircraft". In: *arXiv preprint arXiv:1306.5151* (2013).

[MZ15] Shike Mei and Xiaojin Zhu. "Using Machine Teaching to Identify Optimal Training-Set Attacks on Machine Learners." In: *AAAI*. 2015, pp. 2871–2877.

[NPX+14] Andrew Newell, Rahul Potharaju, Luojie Xiang, and Cristina Nita-Rotaru. "On the Practicality of Integrity Attacks on Document-Level Sentiment Analysis". In: *Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop*. ACM. 2014, pp. 83–93.

[NZ08] Maria-Elena Nilsback and Andrew Zisserman. "Automated flower classification over a large number of classes". In: *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*. 2008.

[PVZ+12] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. "Cats and dogs". In: *2012 IEEE conference on computer vision and pattern recognition*. IEEE. 2012, pp. 3498–3505.

[RDS+15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)*. 2015.

[RHG+15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems (NeurIPS)*. 2015.

[RKB+18] Albert Reuther, Jeremy Kepner, Chansup Byun, Siddharth Samsi, William Arcand, David Bestor, Bill Bergeron, Vijay Gadepally, Michael Houle, Matthew Hubbell, Michael Jones, Anna Klein, Lauren Milechin, Julia Mullen, Andrew Prout, Antonio Rosa, Charles Yee, and Peter Michaleas. "Interactive supercomputing on 40,000 cores for machine learning and data analysis". In: *2018 IEEE High Performance extreme Computing Conference (HPEC)*. IEEE. 2018, pp. 1–6.

[RKH+21] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. "Learning transferable visual models from natural language supervision". In: *arXiv preprint arXiv:2103.00020*. 2021.

[SIE+20] Hadi Salman, Andrew Ilyas, Logan Engstrom, Ashish Kapoor, and Aleksander Madry. "Do Adversarially Robust ImageNet Models Transfer Better?" In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[SKL17] Jacob Steinhardt, Pang Wei W Koh, and Percy S Liang. "Certified Defenses for Data Poisoning Attacks". In: *NIPS*. 2017.

[SMG+20] Krishna Kumar Singh, Dhruv Mahajan, Kristen Grauman, Yong Jae Lee, Matt Feiszli, and Deepti Ghadiyaram. "Don't judge an object by its context: learning to overcome contextual bias". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11070–11078.

[SSS+17] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. "Revisiting unreasonable effectiveness of data in deep learning era". In: *Proceedings of the IEEE international conference on computer vision*. 2017.

[TKH+21] Philipp Terhörst, Jan Niklas Kolf, Marco Huber, Florian Kirchbuchner, Naser Damer, Aythami Morales, Julian Fierrez, and Arjan Kuijper. "A comprehensive study on face recognition biases beyond demographics". In: *arXiv preprint arXiv:2103.01592* (2021).

[TTM19] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. "Label-Consistent Backdoor Attacks". In: 2019.

[UKE+20] Francisco Utrera, Evan Kravitz, N. Benjamin Erichson, Rajiv Khanna, and Michael W. Mahoney. "Adversarially-Trained Deep Nets Transfer Better". In: *ArXiv preprint arXiv:2007.05869*. 2020.

[WAL19] Sherrie Wang, George Azzari, and David B Lobell. "Crop type mapping without field-level labels: Random forest transfer and unsupervised clustering techniques". In: *Remote sensing of environment* 222 (2019), pp. 303–317.

[WPL+17] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M Summers. "Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases". In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*. 2017.

[XEI+20] Kai Xiao, Logan Engstrom, Andrew Ilyas, and Aleksander Madry. "Noise or signal: The role of image backgrounds in object recognition". In: *arXiv preprint arXiv:2006.09994* (2020).

[XHE+10] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. "Sun database: Large-scale scene recognition from abbey to zoo". In: *Computer Vision and Pattern Recognition (CVPR)*. 2010.

[XJB+16] Michael Xie, Neal Jean, Marshall Burke, David Lobell, and Stefano Ermon. "Transfer learning from deep features for remote sensing and poverty mapping". In: *Thirtieth AAAI Conference on Artificial Intelligence*. 2016.

[XXE12] Han Xiao, Huang Xiao, and Claudia Eckert. "Adversarial Label Flips Attack on Support Vector Machines." In: *European Conference on Artificial Intelligence (ECAI)*. 2012.

[ZXY17] Zhuotun Zhu, Lingxi Xie, and Alan Yuille. "Object Recognition without and without Objects". In: *International Joint Conference on Artificial Intelligence*. 2017.

A Experimental Setup

A.1 ImageNet Models

In this paper, we train a number of ImageNet models and transfer them to various datasets in Sections 3 and 5. We mainly use the ResNet-18 architecture all over the paper. However, we study bias transfers using various architectures in Appendix A.4. We use PyTorch's official implementation for these architectures, which can be found here `https://pytorch.org/vision/stable/models.html`.

Training details. We train our ImageNet models from scratch using SGD by minimizing the standard cross-entropy loss. We train for 16 epochs using a Cyclic learning rate schedule with an initial learning rate of 0.5 and learning rate peak epoch of 2. We use momentum of 0.9, batch size of 1024, and weight decay of $5e^{-4}$. We use standard data-augmentation: *RandomResizedCrop* and *RandomHorizontalFlip* during training, and *RandomResizedCrop* during testing. Our implementation and configuration files are available in the attached code.

A.2 Transfer details from ImageNet to downstream image classification tasks

Transfer datasets. We use the image classification tasks that are used in [SIE+20; KSL19], which have various sizes and number of classes. When evaluating the performance of models on each of these datasets, we report the Top-1 accuracy for balanced datasets and the Mean Per-Class accuracy for the unbalanced datasets. See Table 1 for the details of these datasets. For each dataset, we consider two transfer learning settings: *fixed-feature* and *full-network* transfer learning which we describe below.

Table 1: Image classification benchmarks used in this paper. Accuracy metric is the metric we report for each of the dataset across the paper. Some datasets are imbalanced, so we report Mean Per-Class accuracy for those. For the rest, we report Top-1 accuracy.

Dataset	Size (Train/Test)	Classes	Accuracy Metric
Birdsnap [BLW+14]	32,677/8,171	500	Top-1
Caltech-101 [FFP04]	3,030/5,647	101	Mean Per-Class
Caltech-256 [GHP07]	15,420/15,187	257	Mean Per-Class
CIFAR-10 [Kri09]	50,000/10,000	10	Top-1
CIFAR-100 [Kri09]	50,000/10,000	100	Top-1
FGVC Aircraft [MRK+13]	6,667/3,333	100	Mean Per-Class
Food-101 [BGV14]	75,750/25,250	101	Top-1
Oxford 102 Flowers [NZ08]	2,040/6,149	102	Mean Per-Class
Oxford-IIIT Pets [PVZ+12]	3,680/3,669	37	Mean Per-Class
SUN397 [XHE+10]	19,850/19,850	397	Top-1
Stanford Cars [KDS+13]	8,144/8,041	196	Top-1

Fixed-feature transfer. For this setting, we *freeze* the layers of the ImageNet source model[5], except for the last layer, which we replace with a random initialized linear layer whose output matches the number of classes in the transfer dataset. We now train only this new layer for using SGD, with a batch size of 1024 using cyclic learning rate. For more details and hyperparameter for each dataset, please see config files in the attached code.

Full-network transfer. For this setting, we *do not freeze* any of the layers of the ImageNet source model, and all the model weights are updated. We follow the exact same hyperparameters as the fixed-feature setting.

[5]We do not freeze the batch norm statistics, but only the weights of the model similar to Salman et al. [SIE+20].

A.3 Compute and training time

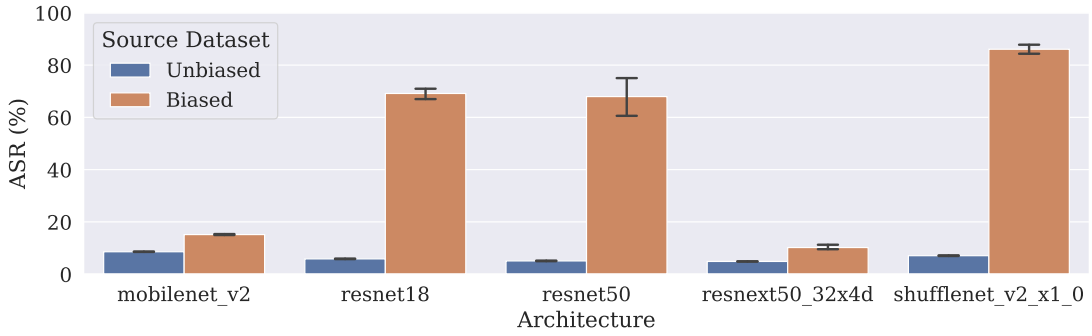
Throughout the paper, we use the FFCV data-loading library to train models fast [LIE+22]. Using FFCV, we can train an ImageNet model, for example, in around 1 hr only on a single V100 GPU. Our experiments were conducted on a GPU cluster containing A100 and V100 GPUs.

A.4 Varying architectures

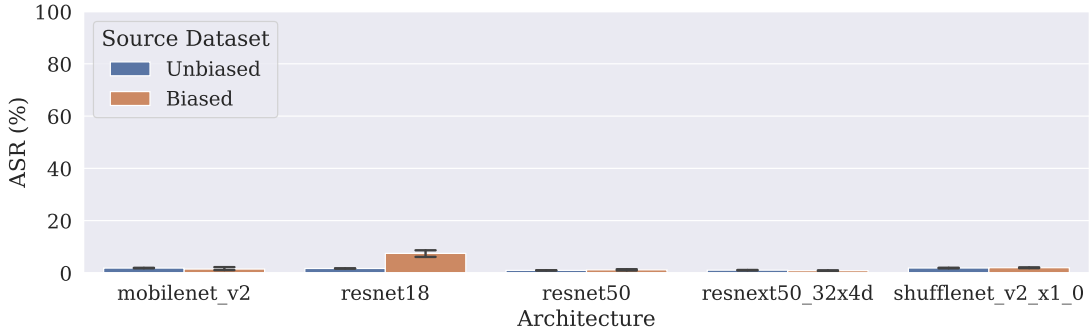
In this section, we study whether bias transfers when applying transfer learning using various architectures. We conduct the basic experiment of Section 3 on several standard architectures from the PyTorch's Torchvision[6].

As in Section 3, we train two versions of each architecture: one on a clean ImageNet dataset, and another on a modified ImageNet dataset containing a backdoor. We use the same hyperparameters as the rest of the paper, except for the batch size, which we set to 512 instead of 1024. The reason we lower the batch size is to fit these models in memory on a single A100 GPU.

Now, we transfer each of these models to a clean CIFAR-10 dataset, and test if the backdoor attack transfers. Similar to the results of the main paper, we notice that backdoor attack indeed transfers in the fixed-feature setting. We note however that for the full-network setting, all architectures other than ResNet-18 (which we use in the rest of the paper) seem to be more robust to the backdoor attack.



(a) CIFAR-10 Fixed-feature



(b) CIFAR-10 Full-network

Figure 9: Backdoor attack (bias) consistently transfers in the fixed-feature setting across various architectures. However, this happens to a lesser degree in the full-network transfer setting.

A.5 MS-COCO

In this section, we provide experimental details for the experiment on MS-COCO in Section 4.1. We consider the binary task of predicting cats from dogs, where there is a strong correlation between dogs and the

[6]These models can be found here https://pytorch.org/vision/stable/models.html

presence of people.

Dataset construction. We create two source datasets which are described in Table 2.

Table 2: The synthetic datasets we create from MS-COCO for the experiment in Section 4.1.

Dataset	*Class: Cat*		*Class: Dog*	
	With People	Without People	With People	Without People
Non-Spurious	0	1000	0	100
Spurious	1000	4000	4000	1000

We then fine-tune models trained on the above source datasets on new images of cats and dogs without people (485 each). We use the cats and dogs from the MS-COCO test set for evaluation.

Experimental details. We train a ResNet-18 with resolution 224×224. We use SGD with momentum, and a Cyclic learning rate. We use the following hyperparameters shown in Table 3:

Table 3: Hyperparameters used for training on the MS-COCO dataset.

Hyperparameter	Value for pre-training	Value for fine-tuning
Batch Size	256	256
Epochs	25	25
LR	0.01	0.005
Momentum	0.9	0.9
Weight Decay	0.00005	0.00005
Peak Epoch	2	2

A.6 CelebA

In this section, we provide experimental details for the CelebA experiments in Section 4.2. Here, the task was to distinguish old from young faces, in the presence of a spurious correlation with gender in the source dataset.

Dataset construction. We create two source datasets shown in Table 4:

Table 4: The synthetic source datasets we create from CelebA for the experiment in Section 4.2.

Dataset	*Class: Young*		*Class: Old*	
	Male	Female	Male	Female
Non-Spurious	2500	2500	2500	2500
Spurious	1000	4000	4000	1000

Due to imbalances in the spurious dataset, the model trained on this dataset struggles on faces of young males and old females. We then fine-tune the source models on the following target datasets (see Table 5), the images of which are disjoint from that in the source dataset.

Due to space constraints, we plotted the results of fixed fine-tuning on Only Women and 80% Women|20% Men in the main paper. Below, we display the results for fixed-feature and full fine-tuning on all 3 target datasets.

Experimental details. We train a ResNet-18 with resolution 224×224. We use SGD with momentum, and a cyclic learning rate. We use the following hyperparameters shown in Table 6:

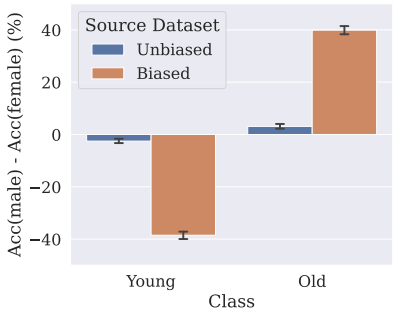
Table 5: The synthetic target datasets we create from CelebA for the experiment in Section 4.2.

Dataset	*Class: Young*		*Class: Old*	
	Male	Female	Male	Female
Only Women	0	5000	0	5000
80% Women|20% Men	1000	4000	1000	4000
50% Women|50% Men	2500	2500	2500	2500

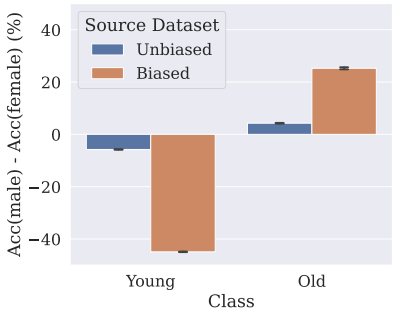
Table 6: Hyperparameters used for training on the CelebA datasets.

Batch Size	Epochs	LR	Momentum	Weight Decay	Peak Epoch
1024	20	0.05	0.9	0.01	5

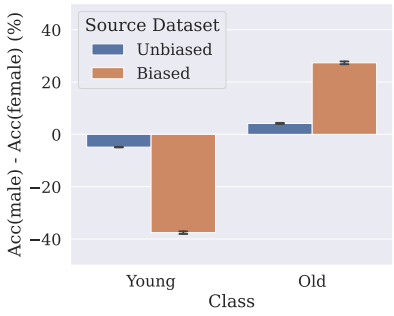
Results. We find that in both the fixed-feature and full-feature fine-tuning settings, the gender correlation transfers from the source model to the transfer model, even though the target task is itself gender balanced. As the proportion of men and women in the target dataset change, the model is either more sensitive to the presence of women, or more sensitive to the presence of men. In all cases, however, the model transferred from the spurious backbone is more sensitive to gender than a model transferred from the non-spurious backbone.



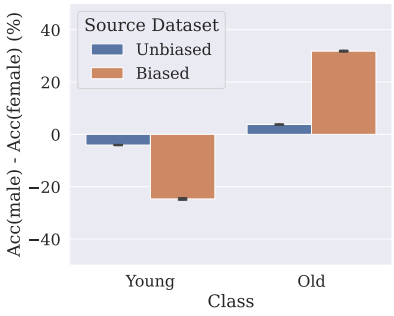
(a) Source Model Accuracy



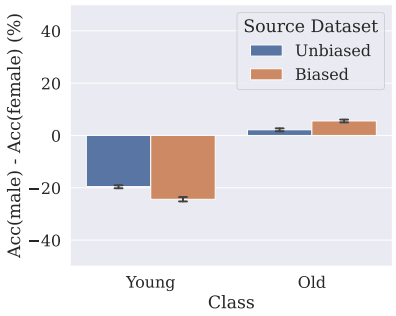
(b) Fixed-Feature Fine-tuning. Target Dataset: Only Women



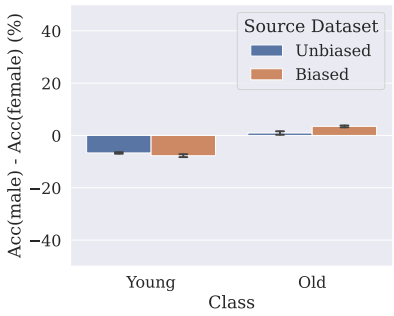
(c) Fixed-Feature Fine-tuning. Target Dataset: 80% Women, 20% Men



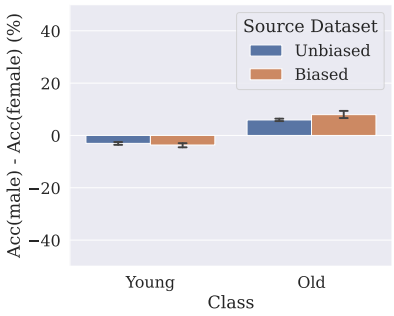
(d) Fixed-Feature Fine-tuning. Target Dataset: 50% Women, 50% Men



(e) Full-Feature Fine-tuning. Target Dataset: Only Women



(f) Full-Feature Fine-tuning. Target Dataset: 80% Women, 20% Men



(g) Full-Feature Fine-tuning. Target Dataset: 50% Women, 50% Men

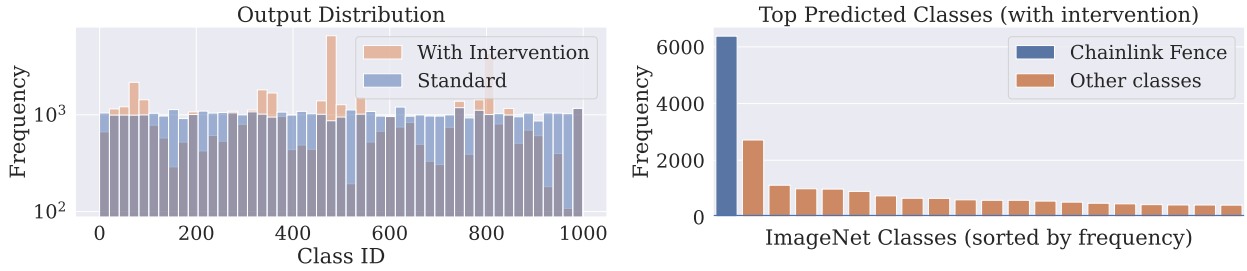
Figure 10: **CelebA Experiment.**We consider transfer from a source dataset that spuriously correlate age with gender — such that old men and young women are overrepresented. We plot the difference in accuracies between male and female examples, and find that the model transferred from a spurious backbone is sensitive to gender, even though the target dataset was itself gender balanced.

B ImageNet Biases

B.1 Chainlink fence bias.

In this section we show the results for the "chainlink fence" bias transfer. We first demonstrate in Figure 11 that the "chainlink fence" bias actually exists in ImageNet. Then in Figures 12, 13, 14, and 15, we show the output distribution—after applying a chainlink fence intervention—of models trained on various datasets either from scratch, or by transferring from the ImageNet model. The from-scratch models are not affected by the chainlink fence intervention, while the ones learned via transfer have highly skewed output distributions.

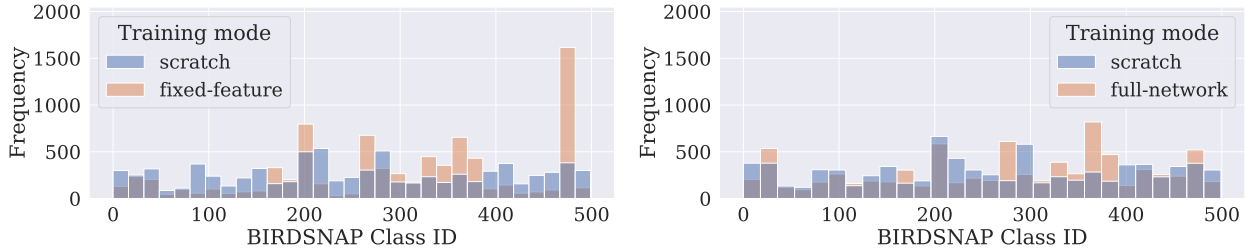
(a) Example images from the "chainlink fence" class in ImageNet.



(b) Shift in ImageNet predicted class distribution after adding a "chainlink fence" intervention, establishing that the bias holds for the source model.

Figure 11: The **chainlink fence** bias in ImageNet.

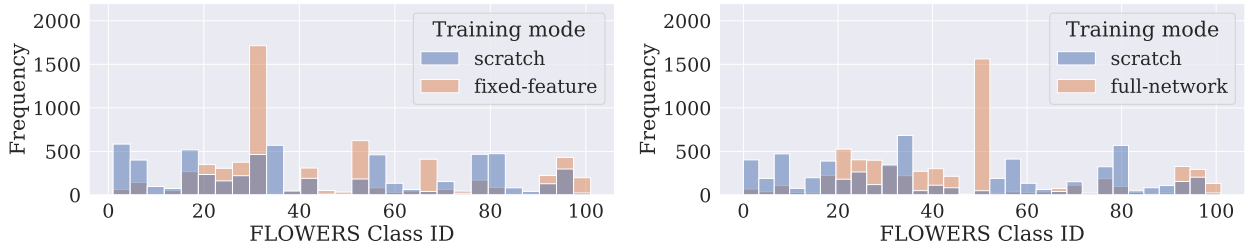
(a) Example Birdsnap images after applying the chain-link fence intervention.



(b) Output distribution of Birdsnap models with a chainlink fence intervention.

Figure 12: The **chainlink fence** bias transfers to *Birdsnap*.

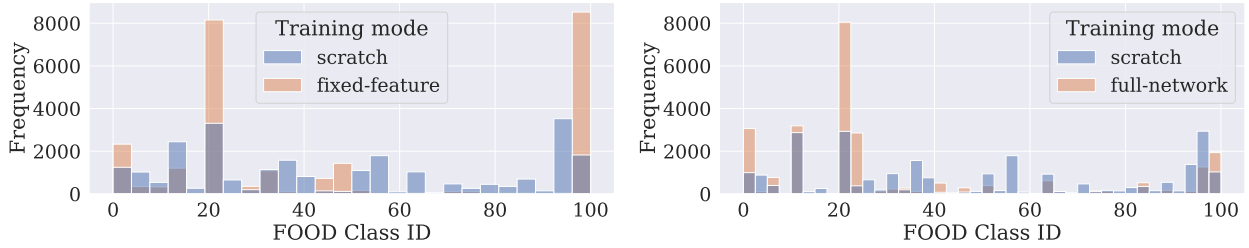
(a) Example Flowers images after applying the chain-link fence intervention.



(b) Output distribution of Flowers models with a chainlink fence intervention.

Figure 13: The **chainlink fence** bias transfers to *Flowers*.

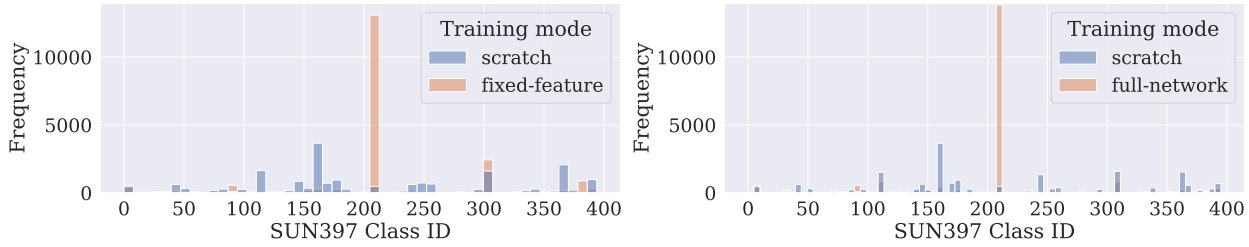
(a) Example Food images after applying the chain-link fence intervention.



(b) Output distribution of Food models with a chainlink fence intervention.

Figure 14: The **chainlink fence** bias transfers to *Food*.

(a) Example SUN397 images after applying the chain-link fence intervention.



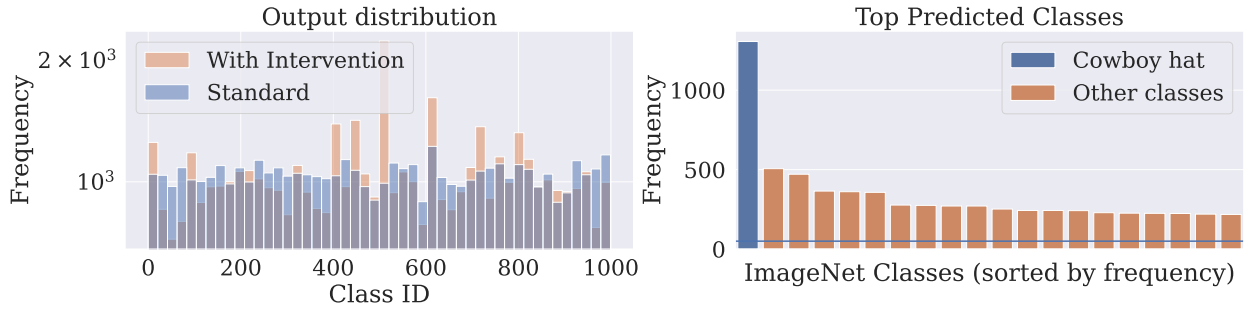
(b) Output distribution of SUN397 models with a chainlink fence intervention.

Figure 15: The **chainlink fence** bias transfers to *SUN397*.

B.2 Hat bias.

In this section we show the results for the "Hat" bias transfer. We first demonstrate in Figure 16 that the "Hat" bias actually exists in ImageNet (shifts predictions to the "Cowboy hat" class). Then in Figure 17, we show the output distribution—after applying a hat intervention—of models trained on CIFAR-10 either from scratch, or by transferring from the ImageNet model. The from-scratch model is not affected by the hat intervention, while the one learned via transfer have highly skewed output distributions.

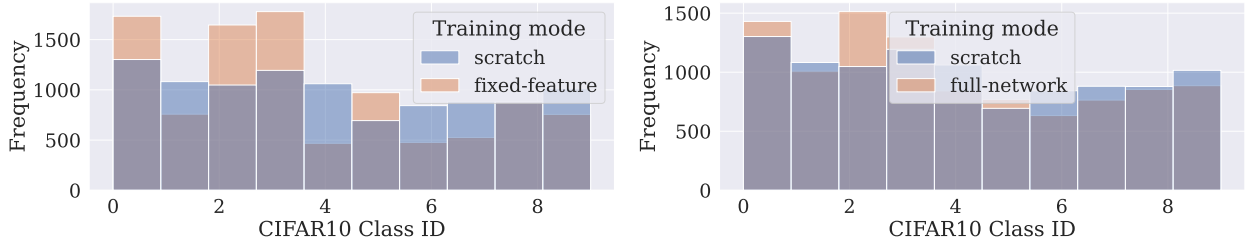
(a) ImageNet images from the class "Cowboy hat".



(b) ImageNet distribution shift after intervention.

Figure 16: The **hat** bias in ImageNet.

(a) Example CIFAR-100 images after applying the "Hat" intervention.

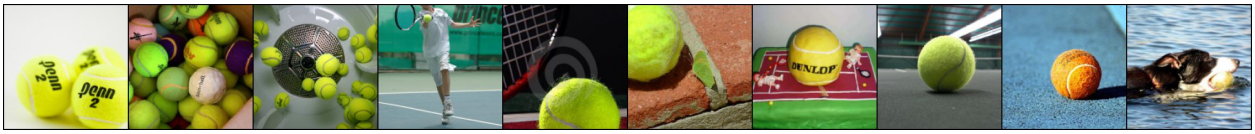


(b) Output distribution of CIFAR-10 models with the Hat intervention.

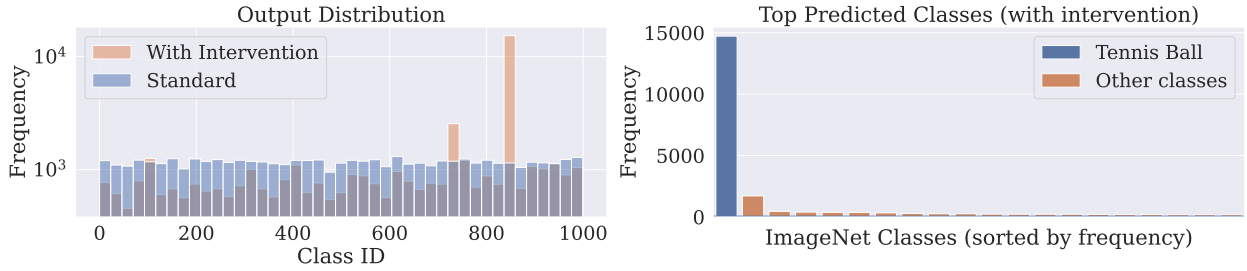
Figure 17: The **hat** bias transfers to *CIFAR-10*.

B.3 Tennis ball bias.

In this section we show the results for the "tennis ball" bias transfer. We first demonstrate in Figure 18 that the "tennis ball" bias actually exists in ImageNet. Then in Figures 19, 20, 21, and 22, we show the output distribution—after applying a tennis ball intervention—of models trained on various datasets either from scratch, or by transferring from the ImageNet model. The from-scratch models are not affected by the tennis ball intervention, while the ones learned via transfer have highly skewed output distributions.

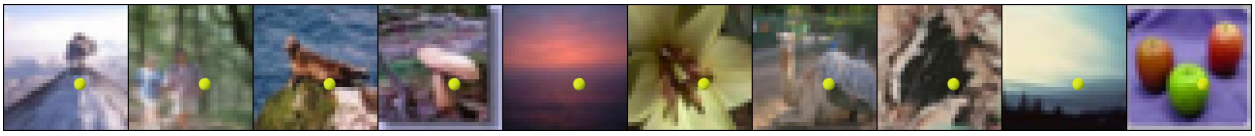


(a) ImageNet images from the class "tennis ball".

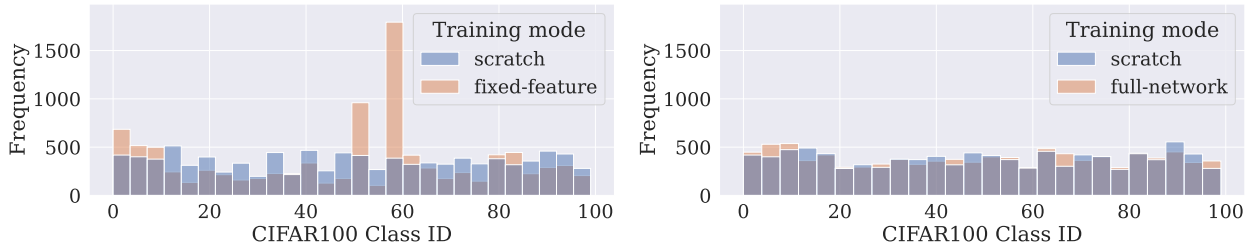


(b) ImageNet distribution shift after intervention.

Figure 18: The **tennis ball** bias in ImageNet.



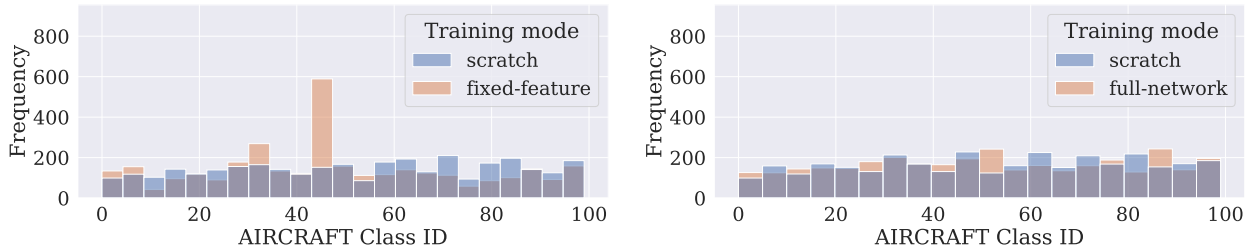
(a) Example CIFAR-100 images after applying the "tennis ball" intervention.



(b) Output distribution of CIFAR-100 models with the tennis ball intervention.

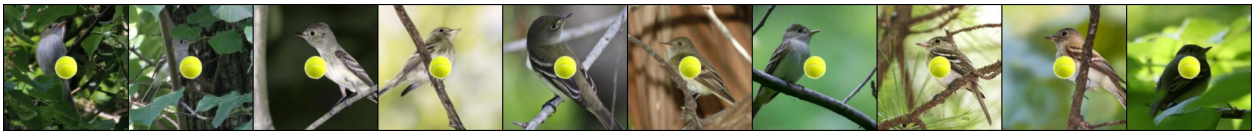
Figure 19: The **tennis ball** bias transfers to *CIFAR-100*.

(a) Example Aircraft images after applying the "tennis ball" intervention.

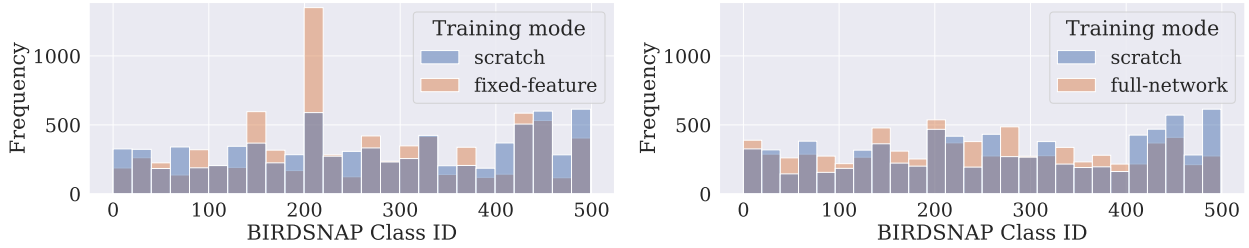


(b) Output distribution of Aircraft models with the tennis ball intervention.

Figure 20: The **tennis ball** bias transfers to *Aircraft*.



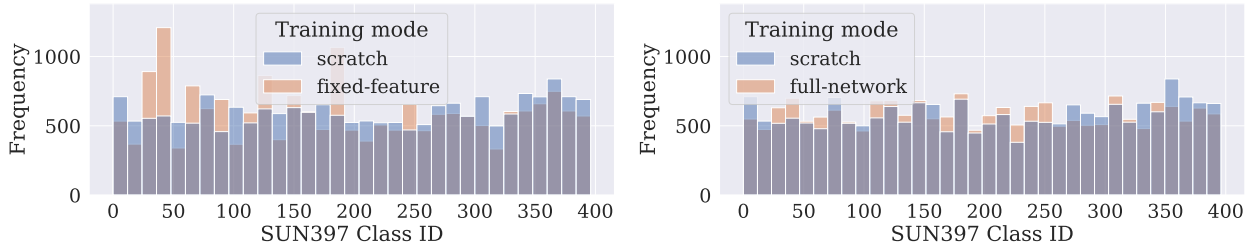
(a) Example Birdsnap after applying the "tennis ball" intervention.



(b) Output distribution of Birdsnap models with the tennis ball intervention.

Figure 21: The **tennis ball** bias transfers to *Birdsnap*.

(a) Example sun397 after applying the "tennis ball" intervention.



(b) Output distribution of SUN397 models with the tennis ball intervention.

Figure 22: The **tennis ball** bias transfers to *SUN397*.