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 (Salman et al., 2020; Kornblith et al., 2019), 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 (Berg et al., 2014)	32,677/8,171	500	Top-1
Caltech-101 (Fei-Fei et al., 2004)	3,030/5,647	101	Mean Per-Class
Caltech-256 (Griffin et al., 2007)	15,420/15,187	257	Mean Per-Class
CIFAR-10 (Krizhevsky, 2009)	50,000/10,000	10	Top-1
CIFAR-100 (Krizhevsky, 2009)	50,000/10,000	100	Top-1
FGVC Aircraft (Maji et al., 2013)	6,667/3,333	100	Mean Per-Class
Food-101 (Bossard et al., 2014)	75,750/25,250	101	Top-1
Oxford 102 Flowers (Nilsback & Zisserman, 2008)	2,040/6,149	102	Mean Per-Class
Oxford-IIIT Pets Parkhi et al. (2012)	3,680/3,669	37	Mean Per-Class
SUN397 (Xiao et al., 2010)	19,850/19,850	397	Top-1
Stanford Cars (Krause et al., 2013)	8,144/8,041	196	Top-1

Fixed-feature transfer. For this setting, we *freeze* the layers of the ImageNet source model[4], 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.

[4]We do not freeze the batch norm statistics, but only the weights of the model similar to (Salman et al., 2020).

A.3 COMPUTE AND TRAINING TIME

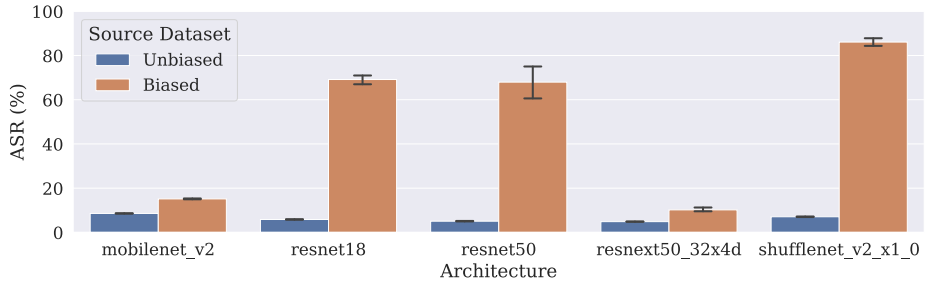
Throughout the paper, we use the FFCV data-loading library to train models fast (Leclerc et al., 2022). Using FFCV, we can train an ImageNet model, for example, in around 1 hr only on a single V100 GPU. Our experiments were conducted on a GPU cluster containing A100 and V100 GPUs.

A.4 VARYING ARCHITECTURES

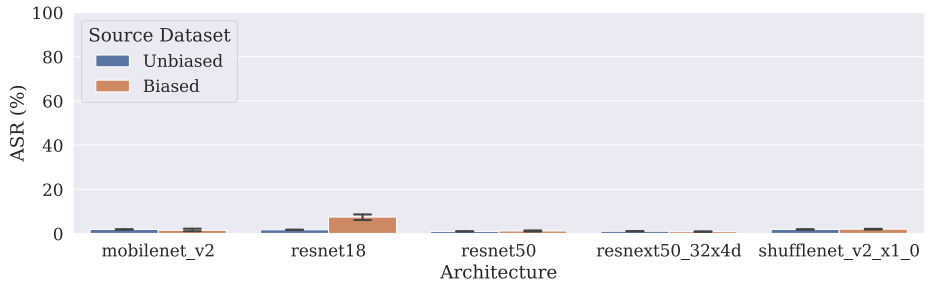
In this section, we study whether bias transfers when applying transfer learning using various architectures. We conduct the basic experiment of Section 3 on several standard architectures from the PyTorch's Torchvision[5].

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.

We further repeat the backdoor experiments for three sizes of Vision Transformer (Figure 10). We find that bias transfer still occurs in both the fixed-feature and full-network settings.

A.5 THE EFFECT OF WEIGHT DECAY IN FULL-NETWORK TRANSFER LEARNING

As mentioned in Section 3.2, we found weight decay to have a significant impact on bias transfer in the full-network transfer learning setting. In particular, increasing weight decay reduces bias transfer. Here, we present a formal explanation of why this happens by studying this within the

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

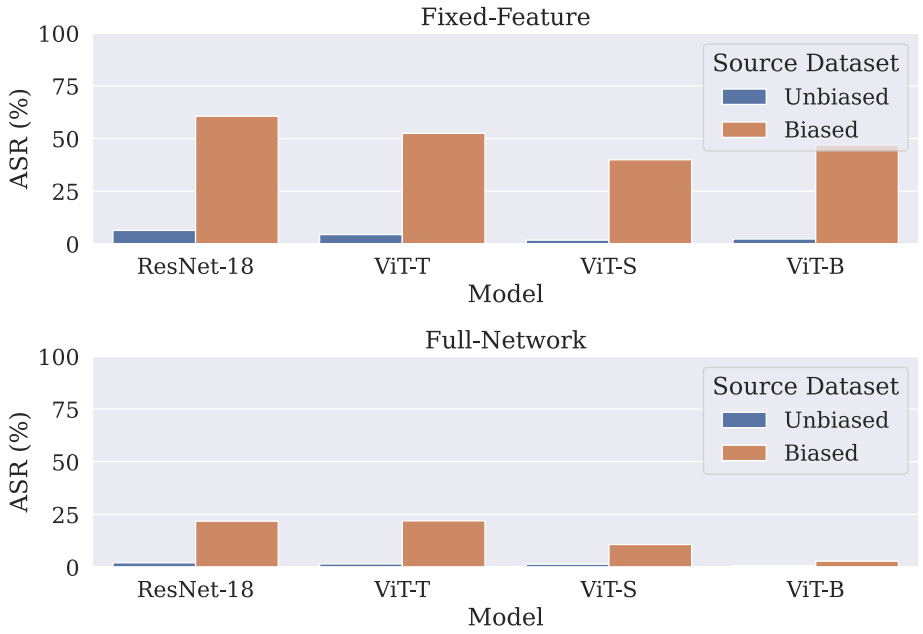


Figure 10: We repeat the backdoor experiments for three sizes of ViT. We find that bias transfer still occurs in both the fixed-feature and full-network settings.

logistic regression example we presented in Section 2. Recall that, following the setup in Section 2, if we transfer a pretrained linear classifier $\boldsymbol{w}_{src}$ to a target dataset $\{(\boldsymbol{x}_i, y_i)\}$, $\boldsymbol{w}_{src}$ is preserved in all directions orthogonal to the span of the $\boldsymbol{x}_i$.

Now what happens if we add ℓ_2 regularization (i.e., weight decay) to the logistic regression problem? As can be easily checked, the gradient updated of the logistic loss now becomes

$$\begin{aligned} \nabla \ell_{\boldsymbol{w}}(\boldsymbol{x}_i, y_i) &= (\sigma(\boldsymbol{w}^\top \boldsymbol{x}_i) - y_i) \cdot \boldsymbol{x}_i + \lambda \boldsymbol{w}, \\ &= (\sigma(\boldsymbol{w}_S^\top \boldsymbol{x}_i) - y_i) \cdot \boldsymbol{x}_i + \lambda(\boldsymbol{w}_S + \boldsymbol{w}_{S'}) \end{aligned} \tag{2}$$

where λ is the regularization strength, $\boldsymbol{w}_S$ and $\boldsymbol{w}_{S'}$ are the projections of $\boldsymbol{w}$ on the span of the target datapoints $\boldsymbol{x}_i$'s, denoted S, and on its complementary subspace, denoted S'. This gradient, as before, restricts the space of updates to those in S. However due to regularization, this gradient drives $\boldsymbol{w}_{S'}$ to zero. Therefore, any planted bias in S' disappears as this subspace collapses to zero with regularization.

Indeed, we observe in practice that as we increase weight decay in the full-network transfer learning regime, bias transfer decreases over various downstream tasks as shown in Figure 11. On the other hand, we find that weight decay does not reduce bias transfer in the fixed feature transfer learning regime, where the weights of the pretrained model are frozen.

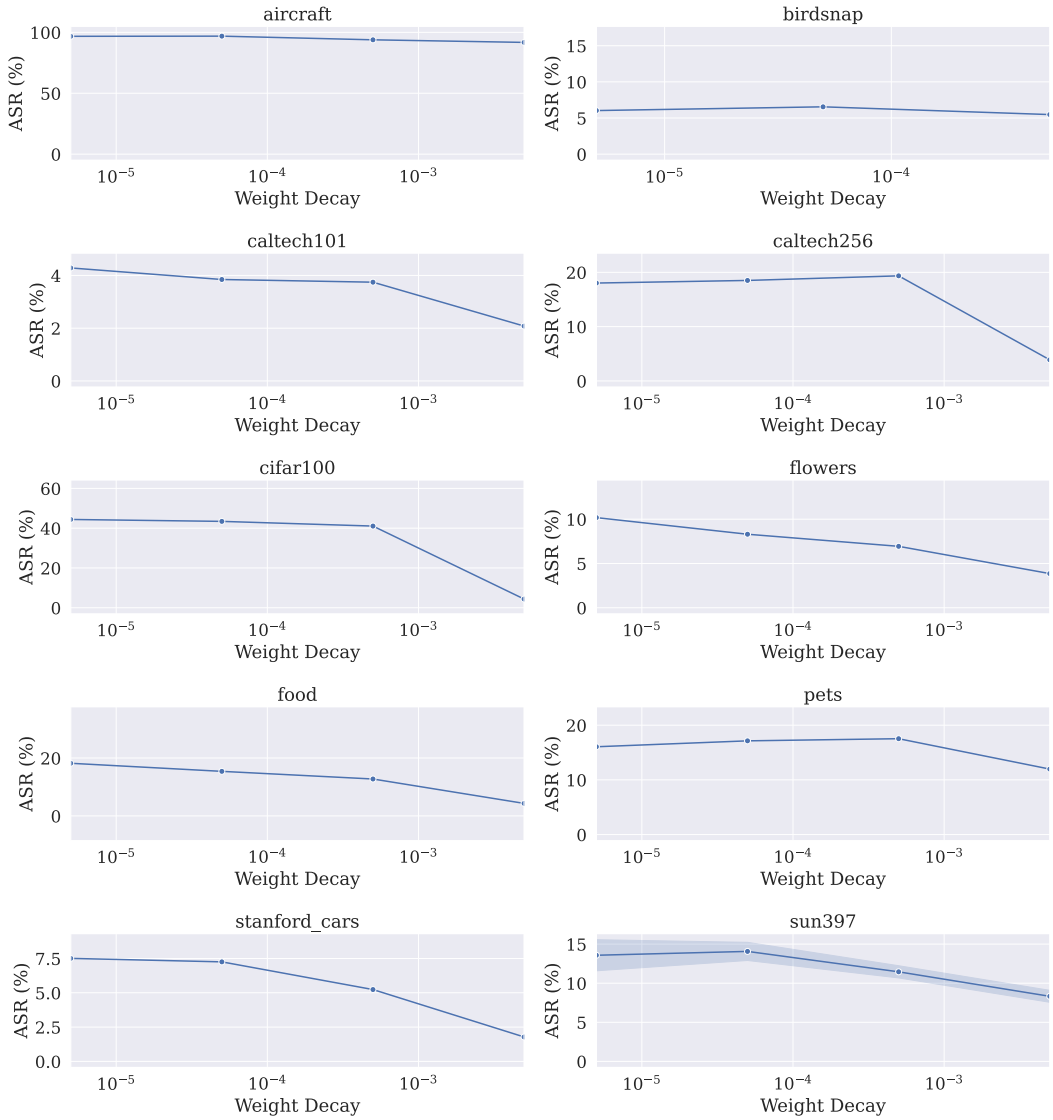


Figure 11: As weight decay increases, the ASR decreases which means bias transfers less across various datasets. We increase weight decay until the clean accuracy on the target dataset significantly deteriorates (see Figure 12). Error regions correspond to standard deviation over five random trials.

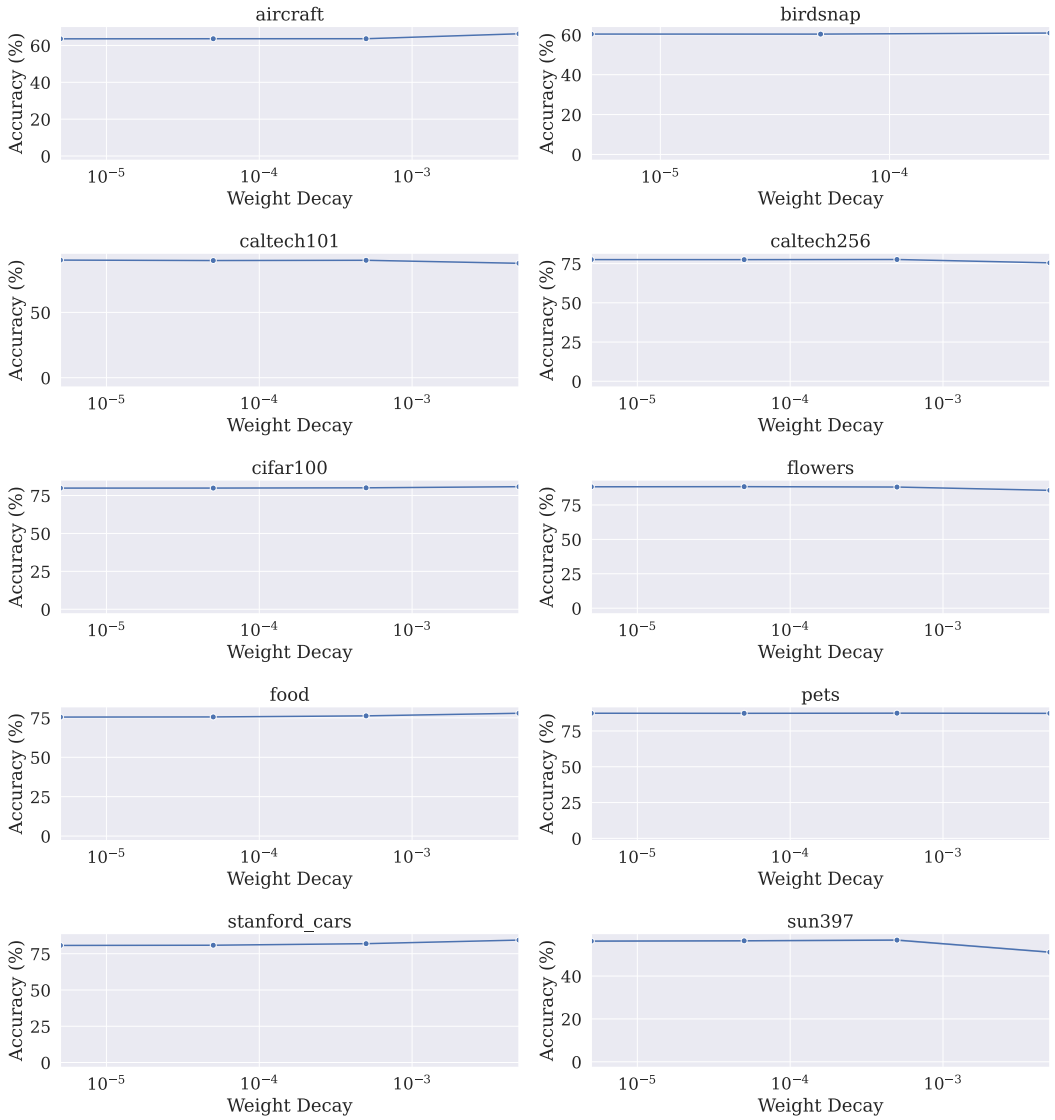
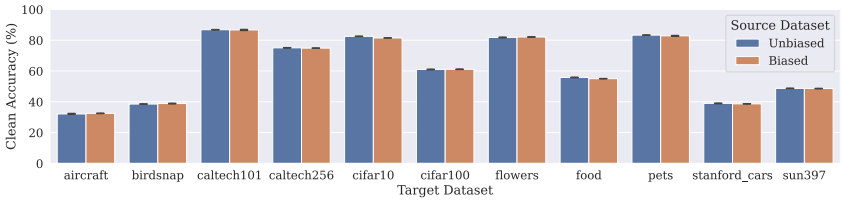


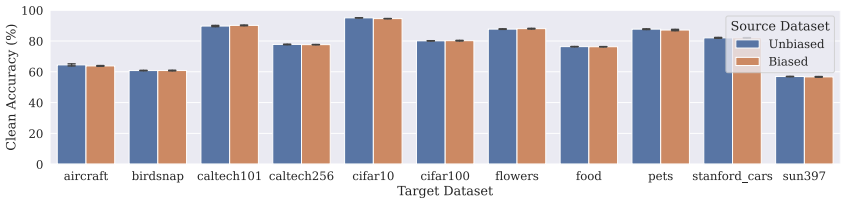
Figure 12: The clean accuracies corresponding to the weight-decay experiment. We increase weight decay as long as the clean accuracy on the target dataset is roughly the same. Error regions (very small) correspond to standard deviation over five random trials.

A.6 CLEAN ACCURACIES FOR EXPERIMENTS OF SECTION 3

In Figure 13, we report the clean accuracies of the transferred models that we use Section 3 on various target datasets. Note how the accuracies of both models pretrained in biased and unbiased source models, for both fixed-feature and full-network settings, are roughly the same. So the discrepancy in ASR reported in the main paper is solely due to bias transfer.



(a) Fixed-feature transfer learning accuracies.

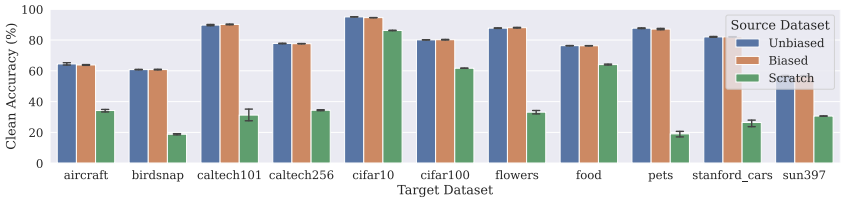


(b) Full-network transfer learning accuracies.

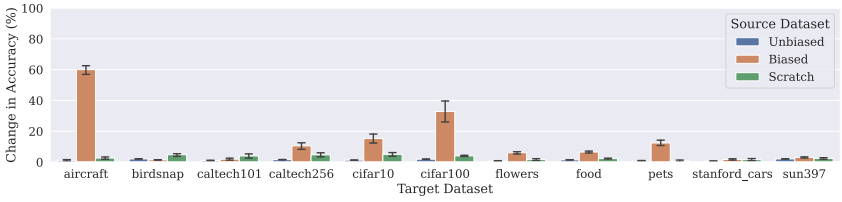
Figure 13: Clean accuracies of the fixed-feature and full-network experiments of Section 3.

A.7 COMPARISON WITH MODELS TRAINED FROM SCRATCH (ADDITIONAL RESULTS TO SECTION 3)

In this section, we add an extra baseline to Figure 3a where we train models from scratch on the various target datasets to check if the yellow square bias already exists in these datasets. In Figure 14a, we plot the accuracies of all the models across all target tasks. Note that since there is a significant difference between the accuracies of the models trained from scratch and those finetuned, ASR is no longer an informative metric to capture the existence of bias. Thus, we measure the change in accuracy after adding the backdoor trigger and report the results in Figure 14b. Indeed, the addition of the yellow square trigger do not significantly change the accuracy of the models trained from scratch reflecting no existing bias in the target datasets.



(a) Full-network transfer learning accuracies.



(b) Full-network transfer learning change in accuracy after adding the backdoor trigger.

Figure 14: Additional baseline ("Scratch") for the experiment of Figure 3a.

A.8 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 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.9 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-feature transfer setting on Only Women and 80% Women|20% Men in the main paper. Below, we display the results for fixed-feature and full-network transfer settings 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:

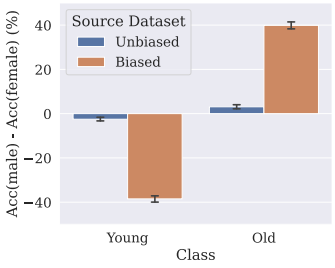
Table 5: The synthetic target datasets we create from CelebA for the experiment in Section 4.2.

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

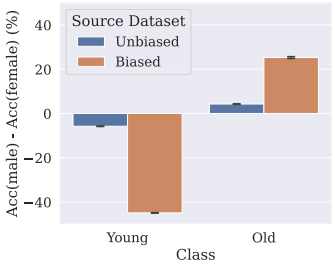
Table 6: Hyperparameters used for training on the CelebA datasets.

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

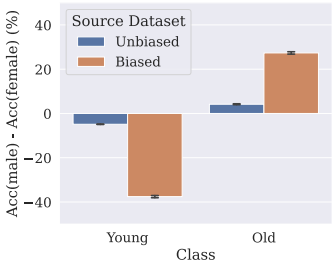
Results. We find that in both the fixed-feature and full-network transfer settings, the gender correlation transfers from the source model to the transfer model, even though the target task is itself gender balanced as shown in Figure 15. As the proportion of men and women in the target dataset change, 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.



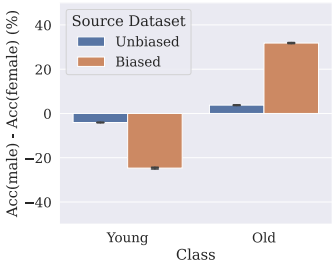
(a) Source Model Accuracy



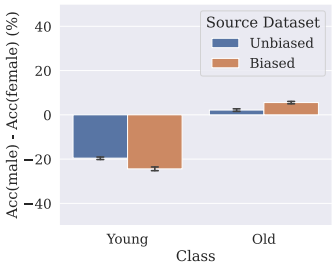
(b) Fixed-Feature transfer. Target Dataset: Only Women



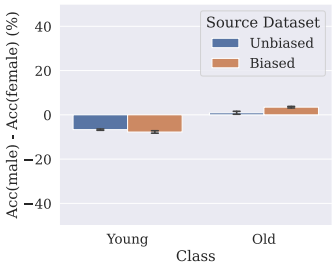
(c) Fixed-Feature transfer. Target Dataset: 80% Women, 20% Men



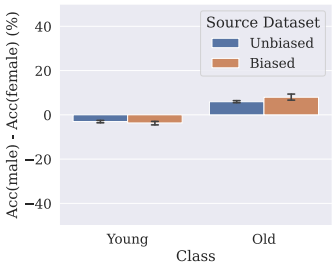
(d) Fixed-Feature transfer. Target Dataset: 50% Women, 50% Men



(e) Full-network transfer. Target Dataset: Only Women



(f) Full-network transfer. Target Dataset: 80% Women, 20% Men



(g) Full-network transfer. Target Dataset: 50% Women, 50% Men

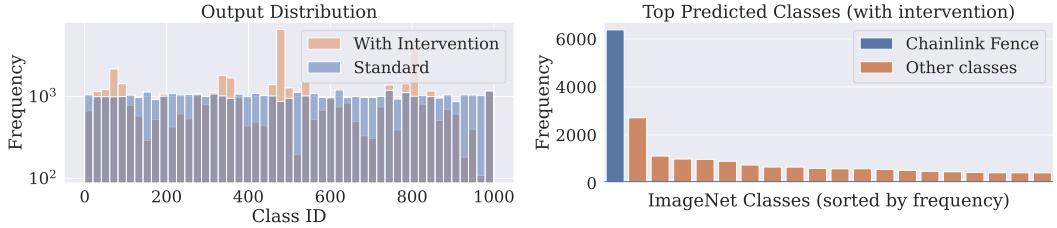
Figure 15: **CelebA Experiment.** We consider transfer from a source dataset that spuriously correlate age with gender — such that old men and young women are overrepresented. We plot the difference in accuracies between male and female examples, and find that the model transferred from a spurious backbone 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 16 that the "chainlink fence" bias actually exists in ImageNet. Then in Figures 17, 18, 19, and 20, 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.

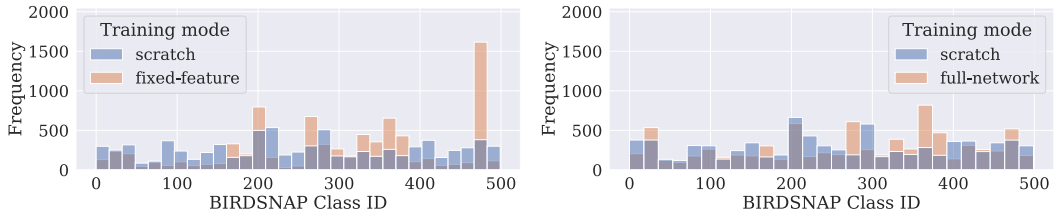
(a) Example images from the "chainlink fence" class in ImageNet.



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

Figure 16: The **chainlink fence** bias in ImageNet.

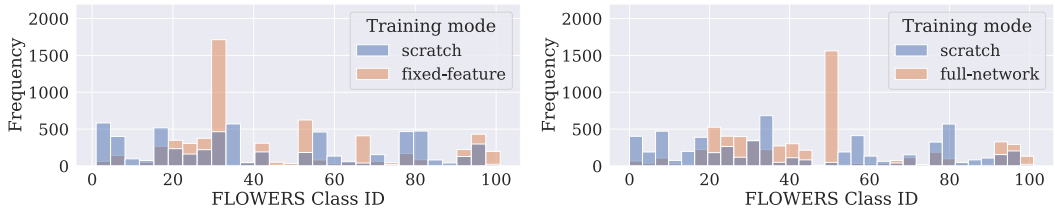
(a) Example Birdsnap images after applying the chain-link fence intervention.



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

Figure 17: The **chainlink fence** bias transfers to *Birdsnap*.

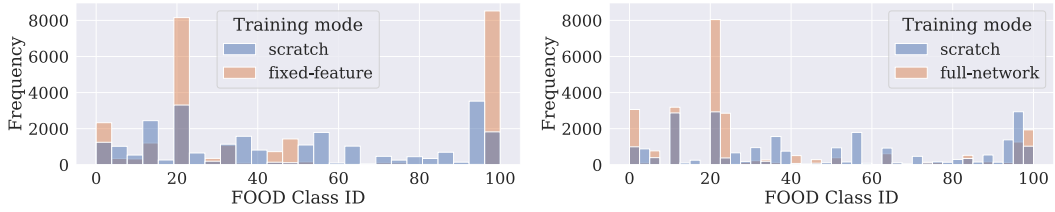
(a) Example Flowers images after applying the chain-link fence intervention.



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

Figure 18: The **chainlink fence** bias transfers to *Flowers*.

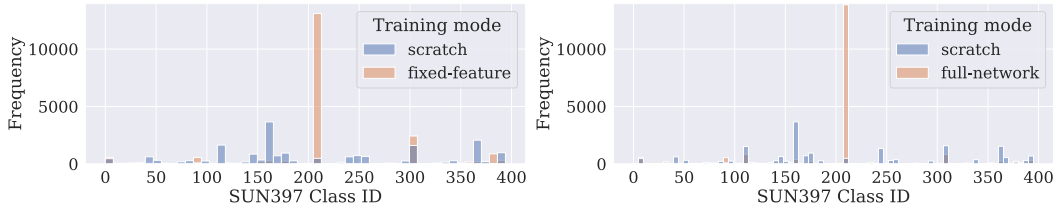
(a) Example Food images after applying the chain-link fence intervention.



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

Figure 19: The **chainlink fence** bias transfers to *Food*.

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



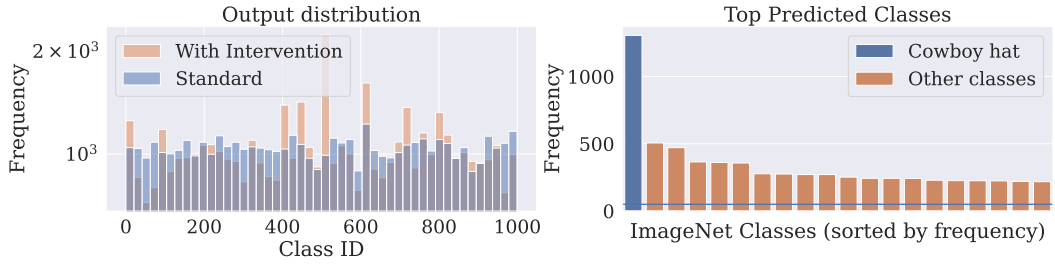
(b) Output distribution of SUN397 models with a chainlink fence intervention.

Figure 20: 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 21 that the "Hat" bias actually exists in ImageNet (shifts predictions to the "Cowboy hat" class). Then in Figure 22, 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.

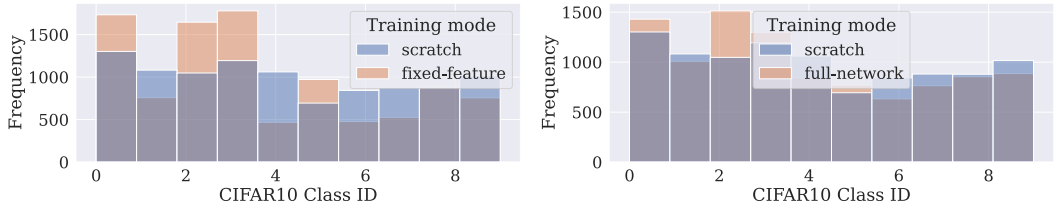
(a) ImageNet images from the class "Cowboy hat".



(b) ImageNet distribution shift after intervention.

Figure 21: The **hat** bias in ImageNet.

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

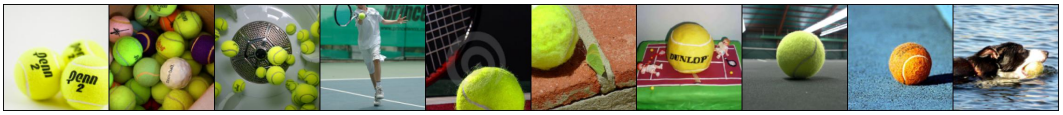


(b) Output distribution of CIFAR-10 models with the Hat intervention.

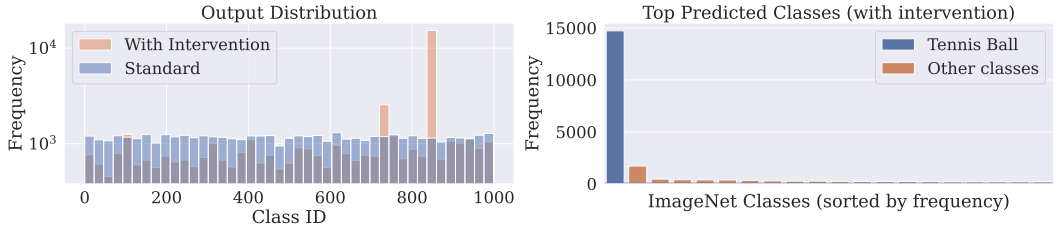
Figure 22: The **hat** bias transfers to *CIFAR-10*.

B.3 TENNIS BALL BIAS.

In this section we show the results for the "tennis ball" bias transfer. We first demonstrate in Figure 23 that the "tennis ball" bias actually exists in ImageNet. Then in Figures 24, 25, 26, and 27, 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.

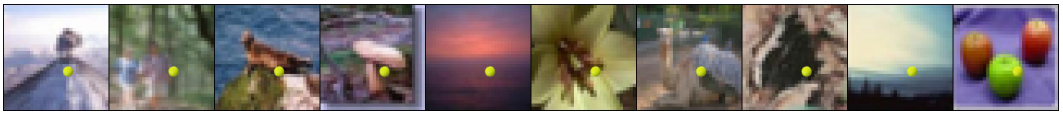


(a) ImageNet images from the class "tennis ball".

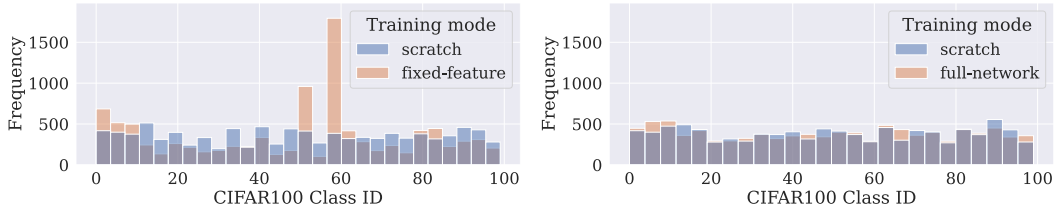


(b) ImageNet distribution shift after intervention.

Figure 23: The **tennis ball** bias in ImageNet.



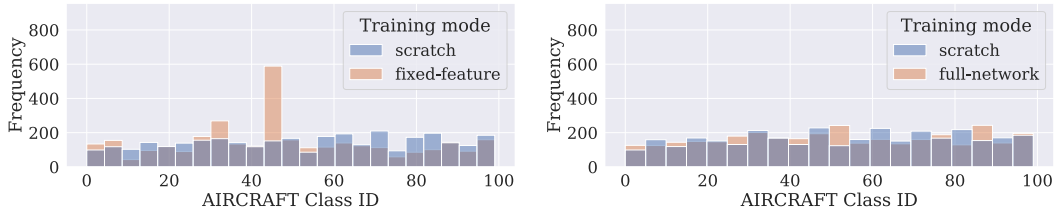
(a) Example CIFAR-100 images after applying the "tennis ball" intervention.



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

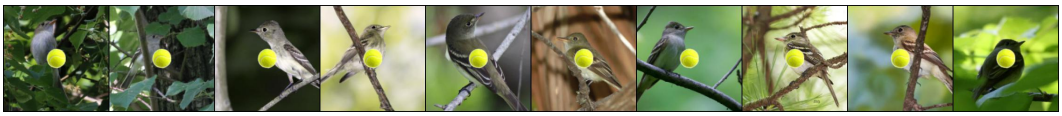
Figure 24: The **tennis ball** bias transfers to *CIFAR-100*.

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

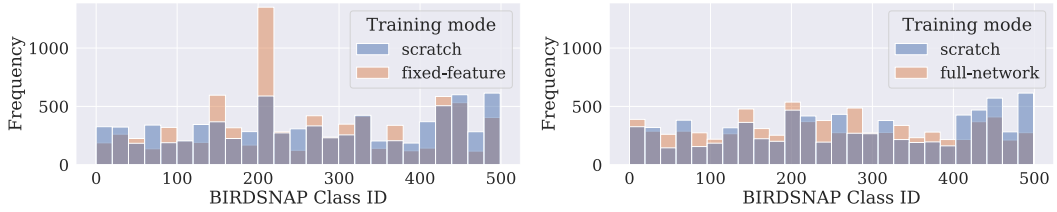


(b) Output distribution of Aircraft models with the tennis ball intervention.

Figure 25: The **tennis ball** bias transfers to *Aircraft*.



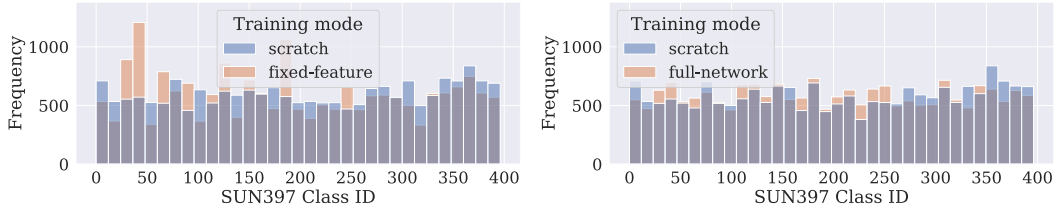
(a) Example Birdsnap after applying the "tennis ball" intervention.



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

Figure 26: 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 27: The **tennis ball** bias transfers to *SUN397*.