# A. Experimental Setup

## A.1. ImageNet Models

In this paper, we train a large number of models on various subsets of ImageNet in order to estimate the influence of each class of ImageNet on the model's transfer performance for multiple downstream tasks. We focus on the ResNet-18 architecture from PyTorch's official implementation found here `https://pytorch.org/vision/stable/models.html`[5].

**Training details.** We fix the training procedure for all of our models. Specifically, we train our models from scratch using SGD to minimize the standard cross-entropy multi-class classification loss. We use a batch size of 1024, momentum of $0.9$, and weight decay of $5 \times 10^{-4}$. The models are trained 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 standard data-augmentation: *RandomResizedCrop* and *RandomHorizontalFlip* during training, and *RandomResizedCrop* during testing. Our implementation and configuration files are attached to the submission.

## A.2. ImageNet transfer to classification datasets

| **Dataset** | **Classes** | **Train Size** | **Test Size** |
|---|---|---|---|
| Birdsnap [2] | 500 | 32,677 | 8,171 |
| Caltech-101 [12] | 101 | 3,030 | 5,647 |
| Caltech-256 [16] | 257 | 15,420 | 15,187 |
| CIFAR-10 [29] | 10 | 50,000 | 10,000 |
| CIFAR-100 [29] | 100 | 50,000 | 10,000 |
| FGVC Aircraft [34] | 100 | 6,667 | 3,333 |
| Food-101 [3] | 101 | 75,750 | 25,250 |
| Oxford 102 Flowers [39] | 102 | 2,040 | 6,149 |
| Oxford-IIIT Pets [40] | 37 | 3,680 | 3,669 |
| SUN397 [46] | 397 | 19,850 | 19,850 |
| Stanford Cars [28] | 196 | 8,144 | 8,041 |

Table 1. Image classification datasets used in this paper.

**Datasets.** We consider the transfer image classification tasks that are used in [27, 42], which vary in size and number of classes. See Table 1 for the details of these datasets. We consider two transfer learning settings for each dataset: *fixed-feature* and *full-network* transfer learning.

**Fixed-feature transfer.** For this setting, we *freeze* the layers of the ImageNet source model[6], 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.

**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.

## A.3. Compute and training time.

We leveraged the FFCV data-loading library for fast training of the ImageNet models [32][7]. Our experiments were run on two GPU clusters: an A100 and a V100 cluster.

---

[5] Our framework is agnostic to the choice of the model's architecture.

[6] For all of our experiments, we do not freeze the batch norm statistics. We only freeze the weights of the model, similar to [42].

[7] Using FFCV, we can train a model on the ImageNet dataset in around 1 hour, and reach ∼63% accuracy

**Training ImageNet models and influence calculation.** We trained 7,540 ImageNet models on random subsets of ImageNet, each containing half of ImageNet classes. On a single V100, training a single model takes around 30 minutes. After training these ImageNet models, we compute the influence of each class as outlined in Algorithm 1. Computing the influences is fast, and takes few seconds on a single V100 GPU.

## A.4. Handpicked baseline details

In our counterfactual experiments in Section 3, we automatically selected, via our framework, the most influential subsets of ImageNet classes for various downstream tasks. We then removed the classes that are detrimental to the transfer performance, and measured the transfer accuracy improvement after removing these classes. The results are summarized in Table 2b.

**What happens if we hand-pick the source dataset classes that are relevant to the target dataset?** Indeed, [38] found that hand-picking the source dataset classes can sometimes boost transfer performance. We validate this approach for our setting using the WordNet hierarchy [35]. Specifically, for each class from the target task, we look up all the ImageNet classes that are either children or parents of this target class. The set of all such ImageNet classes are used as the handpicked most influential classes. Following this manual selection, we train an ImageNet model on these classes, then apply transfer learning to get the baseline performance that we report in Table 2b.

## A.5. Convergence Analysis

We compute our class influence values using 7,540 source models, each of which were trained using 500 randomly chosen ImageNet classes. How many models do we actually need to compute our transfer influences?

**Counterfactual Experiment** To first analyze this question, we re-run our counterfactual experiment in Section 3 when using a smaller number of models to compute transfer influences (Figure 8). While using the full number of models performs the best, we get meaningful transfer influences when computing with both 4000 and 1000 models. In both cases, removing the most negatively influential classes boosts transfer learning performance over using the entire source dataset, while removing the most positively influential classes drops transfer learning performance over the random baseline.

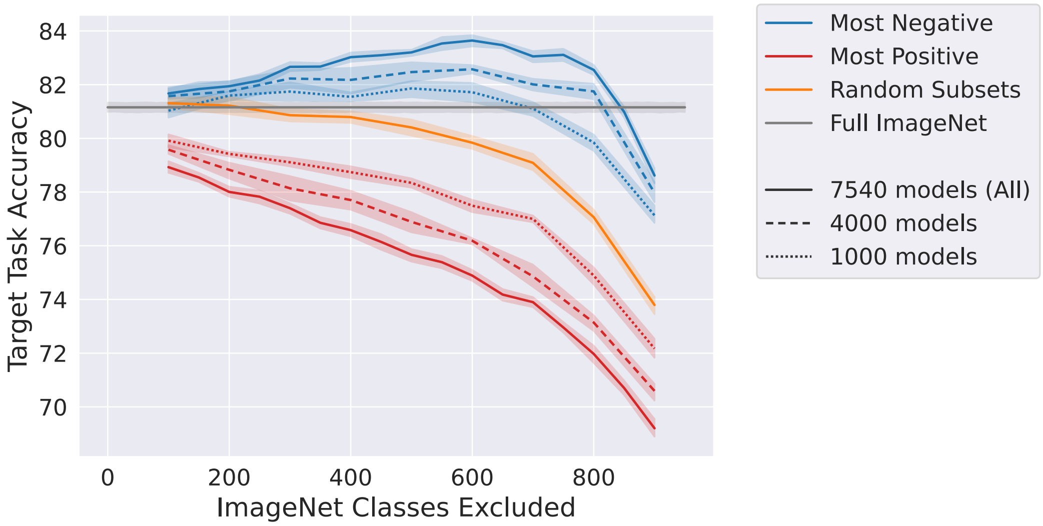


Figure 8. We replicate the counterfactual experiment in Section 3 Figure 2a using 1000 and 4000 source models for computing the transfer influences.

**Bootstrap Analysis** In order to analyze the convergence of the transfer influences, we track the standard deviation of the influence values after bootstrap resampling.

We consider the ImageNet → CIFAR-10 transfer setting with fixed-feature fine-tuning. Given $N$ models, we randomly sample, with replacement, $N$ models to recompute our transfer influences. Specifically, we evaluate the overall transfer influences (i.e., the influence value of each ImageNet class averaged over all target examples). We perform this resampling 500 times, and measure the standard deviation of the computed overall transfer influence value for each class over these 500 resamples.

Below, we plot this standard deviation (averaged over the 1000 classes) for various number of models $N$. We find that the standard deviation goes down as more models are used, indicating that our estimate of the influence values has less variance. This metric roughly plateaus by the time we are using 7000 models.

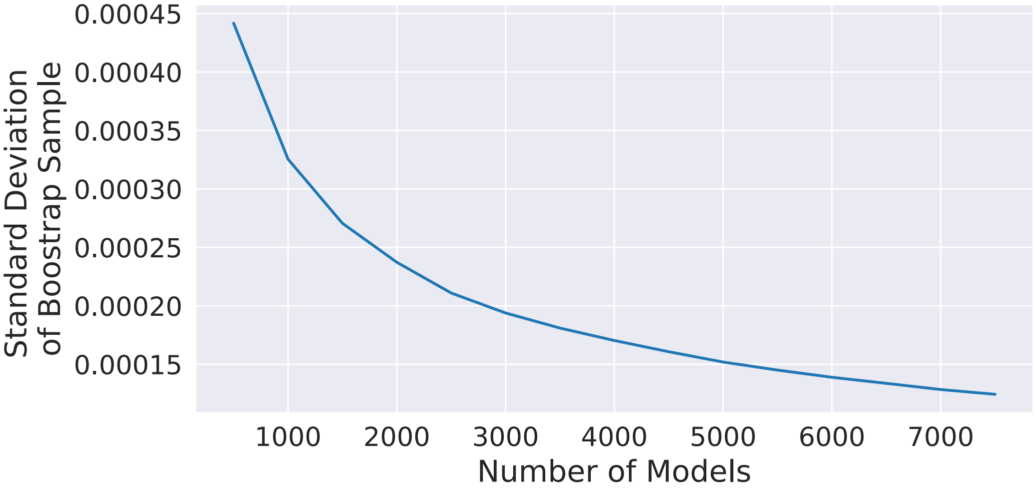


Figure 9. Standard deviation of the overall influence values (averaged over classes) after bootstrap resampling for various numbers of models.

# B. Variants of Computing Influences

## B.1. Variations of targets for computing transfer influences

In our paper, we used the softmax output of the groundtruth class as the target for our influence calculation. What happens if we use a different target? We compare using the following types of targets.

- Softmax Logits: the softmax output of the groundtruth class
- Is Correct: the binary value of whether the image was predicted correctly
- Raw Margins: the difference in raw output between the correct class and the most confidently predicted incorrect class
- Softmax Margins: the same as raw margins, but use the output after softmax

In Figure 10, we replicate the counterfactual experiment from the main paper in Figure 2b using these different targets. Specifically (over 2 runs), we rank the overall influence of the ImageNet classes on CIFAR-10 for fixed-feature transfer. We then remove the classes in order most most or least influence.

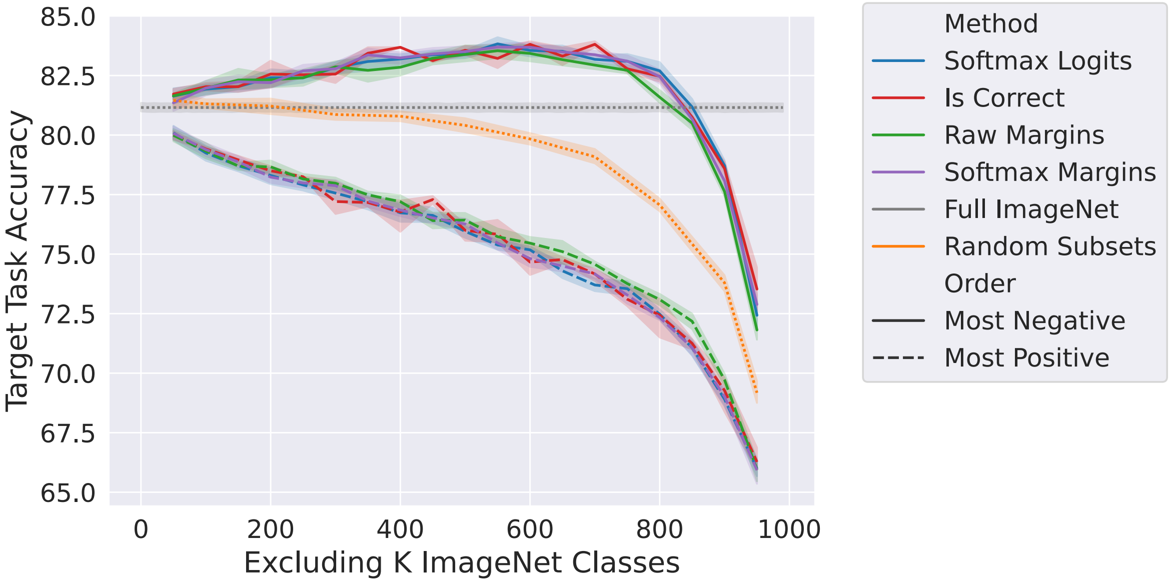


Figure 10. Target task accuracies after removing the most positively or negatively influential ImageNet classes from the source dataset with various influence targets.

We find that our method is not particularly sensitive to the individual target used. We found that using the softmax logits provided the highest benefit when removing negative influencers, and thus used that target for the reset of our experiments.

**Datamodels vs. Influences.** Datamodels [19] is another method that, similar to influences, seeks to compute the importance of a training point on a test set prediction. Specifically, instead of computing the difference in the expected accuracy of the model when a training point is removed, the method fits a linear model that, given a binary vector that denotes the composition of the training dataset, predicts the raw margin (i.e., the difference in raw output between the correct class and the most confidently predicted incorrect class). The importance of each training point is then the coefficient of the linear model for that particular example.

We adapt this method to our framework by training a linear model with ridge regression to predict the softmax output of the transfer model on the target images given a binary vector that denotes which source classes were included in the source dataset. However, we find that datamodels were more effective for computing example-based values (see Appendix D).

In Figure 11, we compare using influence values (as described in the main paper) to using these adapted datamodels. Specifically (over 5 runs), we rank the overall importance of the ImageNet classes on CIFAR-10 for fixed-feature transfer using influences or datamodels. We then remove the classes in order most most or least influence. We find that our framework is not sensitive to the choice of datamodels or influences. However, influences performed marginally better in this counterfactual experiment, so we used influences for all other experiments in this paper.

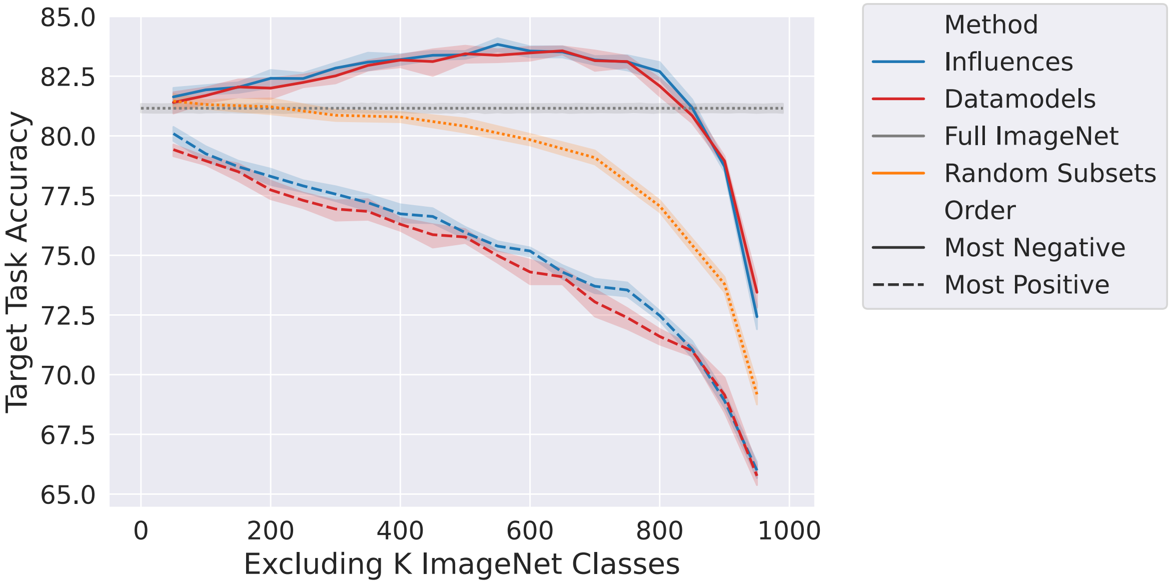


Figure 11. Target task accuracies after removing the most positively or negatively influential ImageNet classes from the source dataset using datamodels or influences.

## C. Full Counterfactual Experiment

In this section, we display the full results for the counterfactual experiment in the main paper (Figure 2b). Specifically, for each target task, we display the target task accuracies after removing the most positive (top) and negative (bottom) influencers from the dataset over 10 runs. We find that our results hold across datasets.

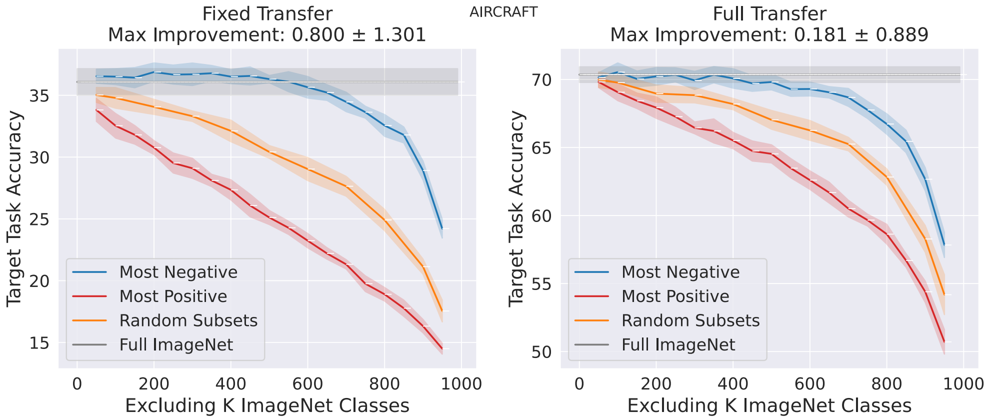


Figure 12. AIRCRAFT

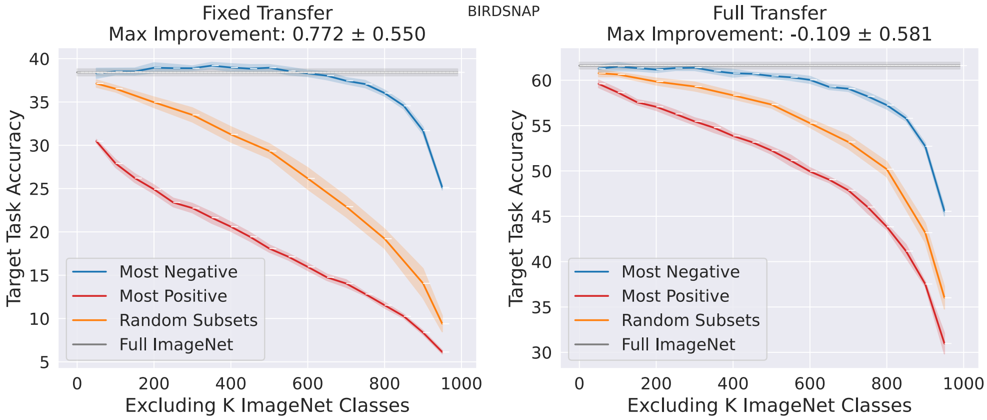


Figure 13. BIRDSNAP

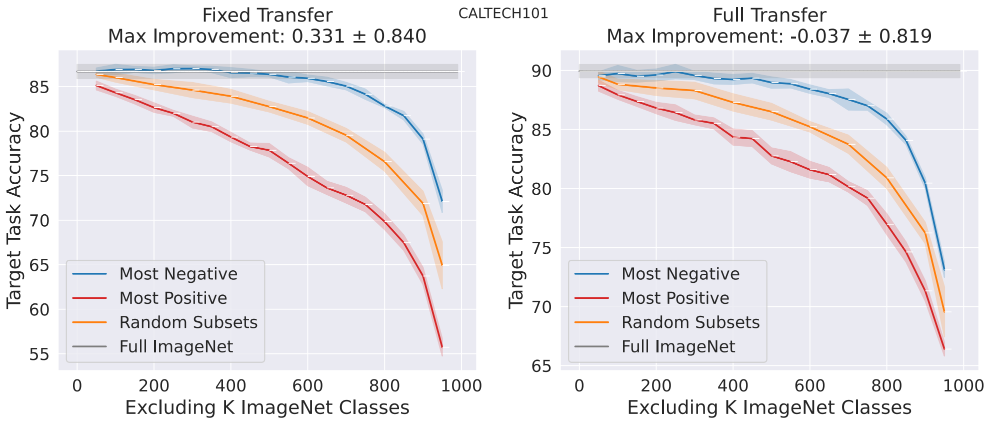


Figure 14. CALTECH101

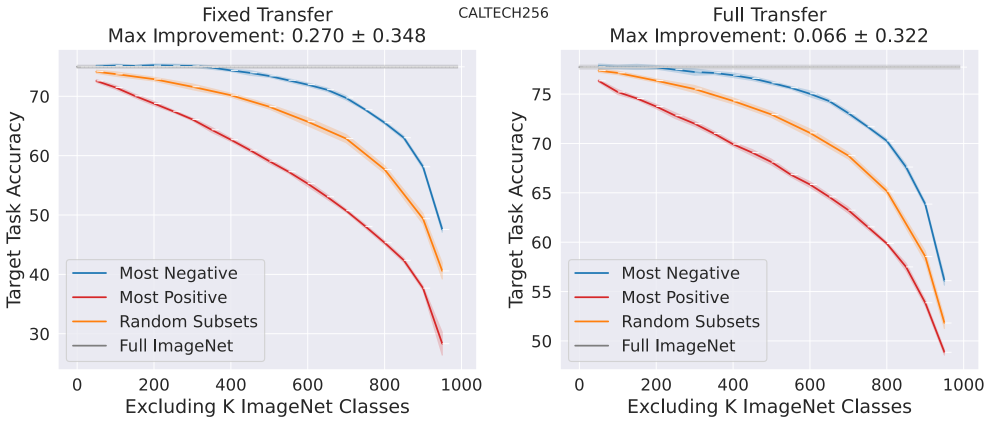


Figure 15. CALTECH256

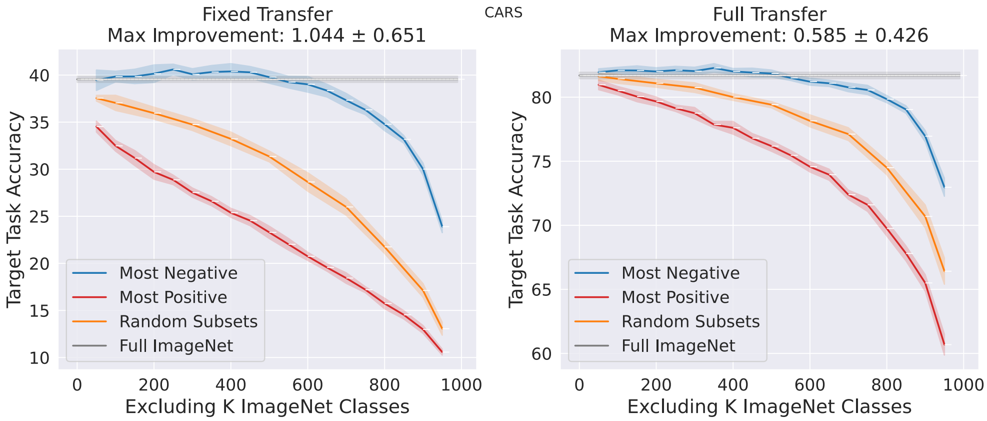


Figure 16. CARS

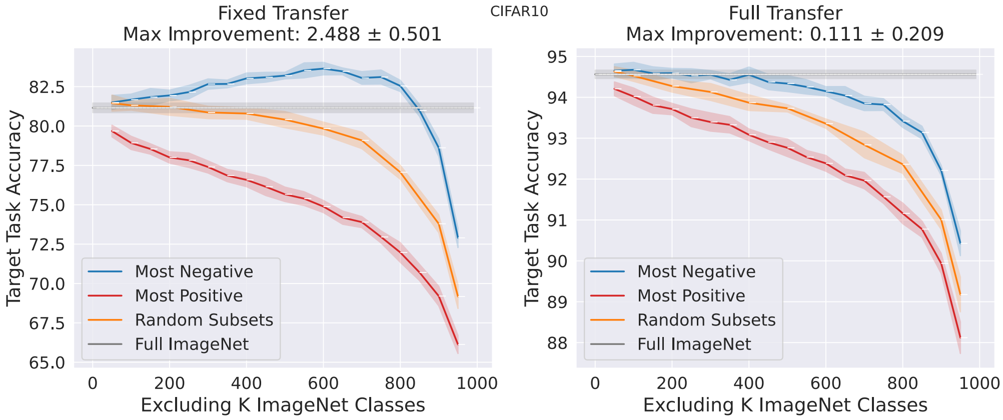


Figure 17. CIFAR10

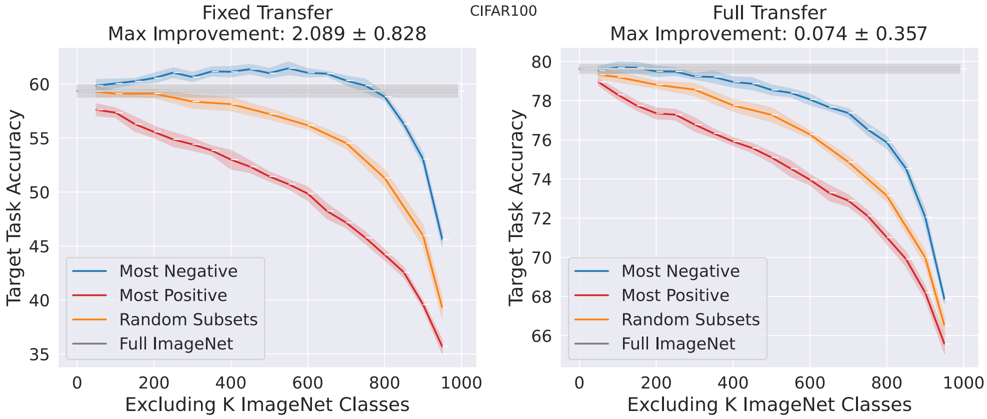


Figure 18. CIFAR100

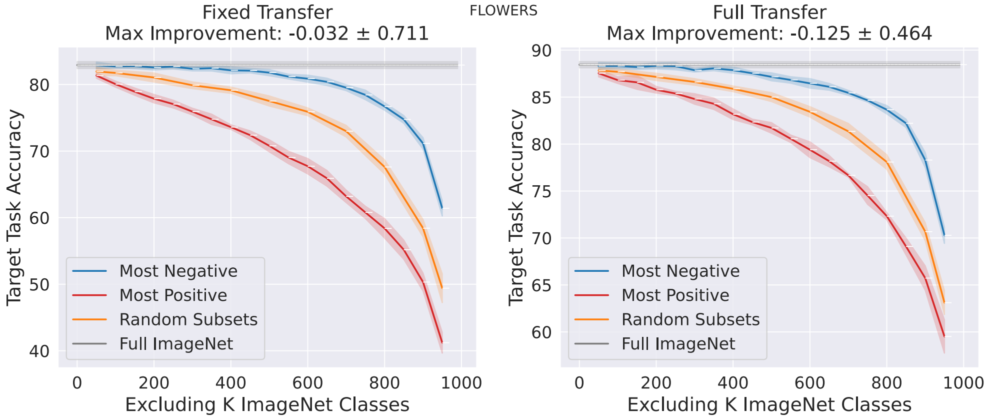


Figure 19. FLOWERS

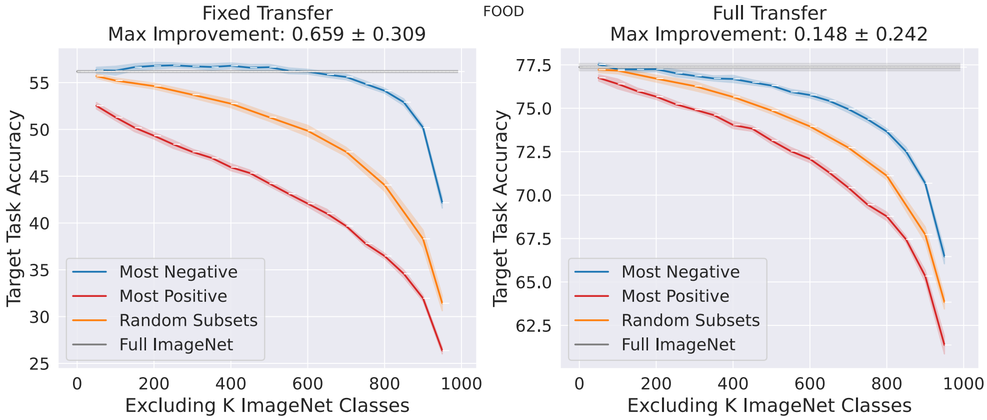


Figure 20. FOOD

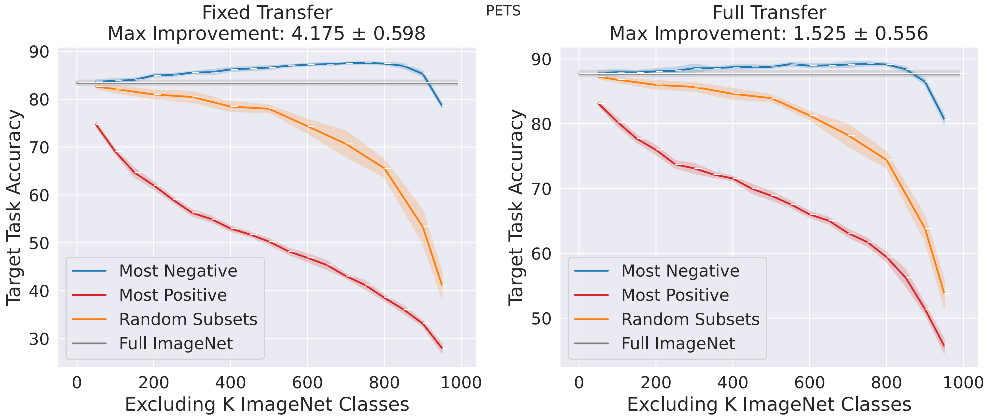


Figure 21. PETS

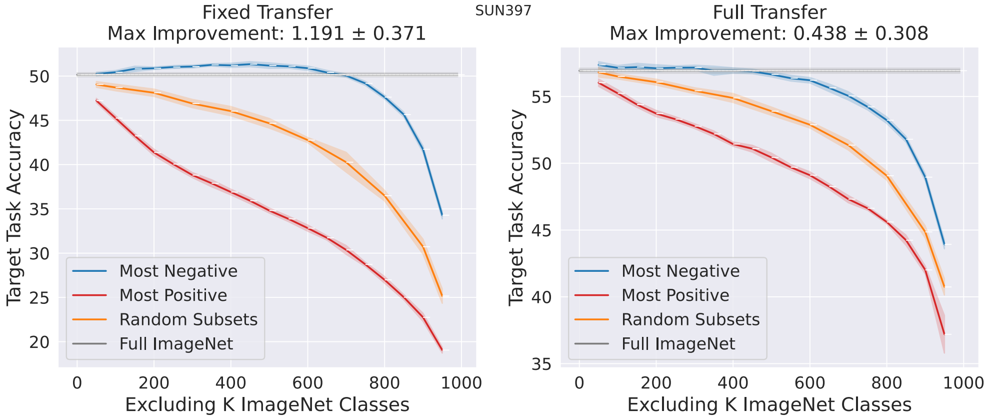
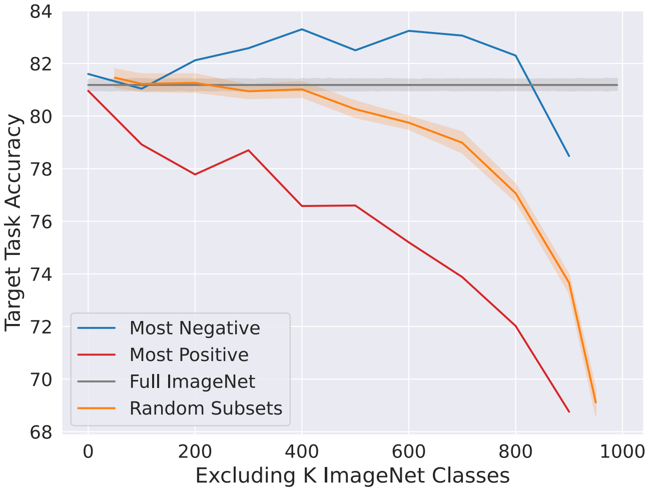


Figure 22. SUN397

## C.1. Performing counterfactual experiments with a validation set

In this section, we replicate the experiment in Figure 2a of the main paper. Here, we split the original CIFAR-10 test set into half, where the first half is the validation set and the second is the test set.

We compute our overall class influences on the validation set, and evaluate the accuracy of removing the top and bottom source classes on the held out test set. The results match Figure 2a. Thus, the transfer influences generalize beyond the specific target examples used to compute the influences.



# D. Adapting our Framework to Compute the Effect of Every Source Datapoint on Transfer Learning

We have presented in the main paper how to compute the influences of every class in the source dataset on the predictions of the model on the target dataset. In that setup, we demonstrated multiple capabilities of our framework, such as improving overall transfer performance, detecting particular subpopulations in the target dataset, etc. Given the wide range of capabilities class-based influences provide, one natural question that arises: Can we compute the influence of every source datapoint on the predictions of the model on the target dataset? Furthermore, what do these influences tell us about the transfer learning process?

Mathematically, the computation of example-based influences (i.e., the influence of every source datapoint) is very similar to the computation of class-based influences. Specifically, to compute example-based influences, we start by training a large number of models on different subsets of the source datapoints (as opposed to source classes for class-based influences). Next, we estimate the influence value of a source datapoint $s$ on a target example $t$ as the expected difference in the transfer model's performance on example $t$ when datapoint $s$ was either included or excluded from the source dataset:

$$\text{Infl}[s \rightarrow t] = \mathbb{E}_S \left[ f(t;S) \mid s \in S \right] - \mathbb{E}_S \left[ f(t;S) \mid s \notin S \right] \tag{2}$$

where $f(t;S)$ is the softmax output of a model trained on a subset $S$ of the source dataset. Similar to class-based influences, a positive (resp. negative) influence value indicates that including the source datapoint $s$ improves (resp. hurts) the model's performance on the target example $t$.

While example-based influences provide some insights about the transfer process, we found that—in this regime—datamodels [19] provide cleaner results and better insights. Generally, influences and datamodels measure similar properties: the effect of the source datapoints on the target datapoints. For a particular target datapoint $t$, we measure the effect of every source datapoint $s$ with datamodels by solving a regression problem. Specifically, we train a large number of models on different subsets of the source dataset. For every model $f_i$, we record 1) a binary mask $\mathbb{1}_{\mathcal{S}_i}$ that indicates which source datapoints were included in the subset $\mathcal{S}_i$ of the source dataset, and 2) the transfer performance $f_i(t;\mathcal{S}_i)$ of the model $f_i$ on the target datapoint $t$ after fine-tuning on the target dataset. Following the training and the fine-tuning stages, we fit a linear model $g_w$ that predicts the transfer performance $f(t;\mathcal{S})$ from a random subset $\mathcal{S}$ of the source dataset as follows: $f(t;\mathcal{S}) \approx g_w(\mathbb{1}_{\mathcal{S}}) = w^\top \mathbb{1}_{\mathcal{S}}$. Given this framework, $w = (w_1, w_2, \ldots, w_L)$ measures the effect of every source datapoint $s$ on the target datapoint $t$[8]. We present the overall procedure in Algorithm 2.

**Algorithm 2** Example-based datamodels estimation for transfer learning.

```
Require: source dataset $\mathcal{S} = \cup_{l=1}^{L} s_l$ (with $L$ datapoints), a target dataset $\mathcal{T} = (t_1, t_2, \cdots, t_n)$, training algorithm $\mathcal{A}$, subset ratio $\alpha$, number of models $m$
Sample $m$ random subsets $S_1, S_2, \cdots, S_m \subset \mathcal{S}$ of size $\alpha \cdot |\mathcal{S}|$:
for $i \in 1$ to $m$ do
    Train model $f_i$ by running algorithm $\mathcal{A}$ on $S_i$
end for
Fine-tune $f_i$ on the training target dataset
for $j \in 1$ to $n$ do
    Collect datamodels training set $\mathcal{D}_j = \{(\mathbb{1}_{\mathcal{S}_i}, f_i(t_j, \mathcal{S}_i))\}_{i=1}^{m}$
    Compute $w_j$ by fitting LASSO on $\mathcal{D}_j$
end for
return $w_j \;\; \forall \;\; j \in [n]$
```

---

[8] To estimate the datamodels, we train 71,828 models on different subsets of the source dataset.

# E. Omitted Results

## E.1. Per-class influencers

We display for the ImageNet → CIFAR-10 the top (most positive) and bottom (most negative) influencing classes for each CIFAR-10 class. This is the equivalent to the plot in Figure 3 in the main paper.

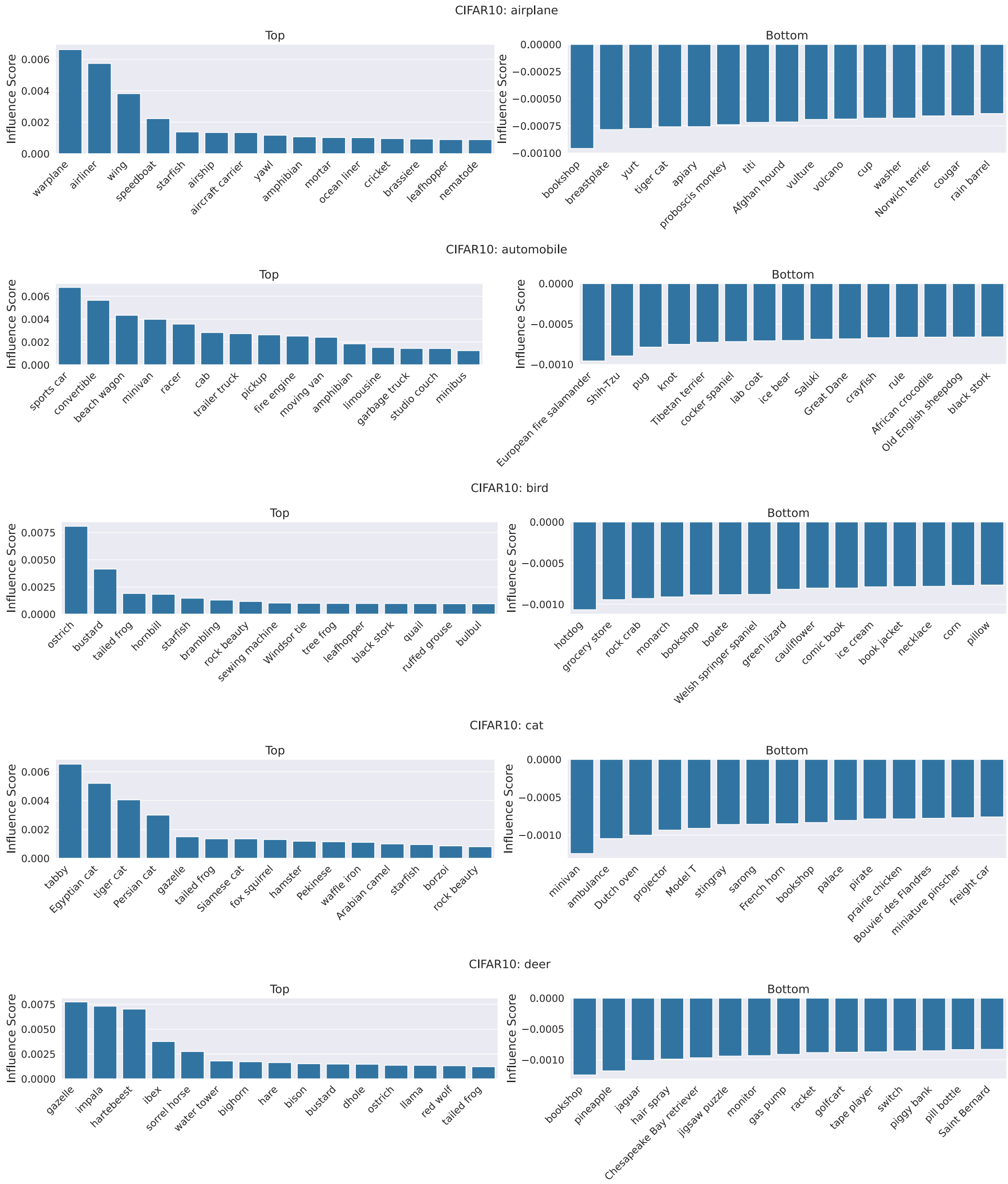


Figure 23. Top and bottom influencing ImageNet classes for all CIFAR-10 classes.

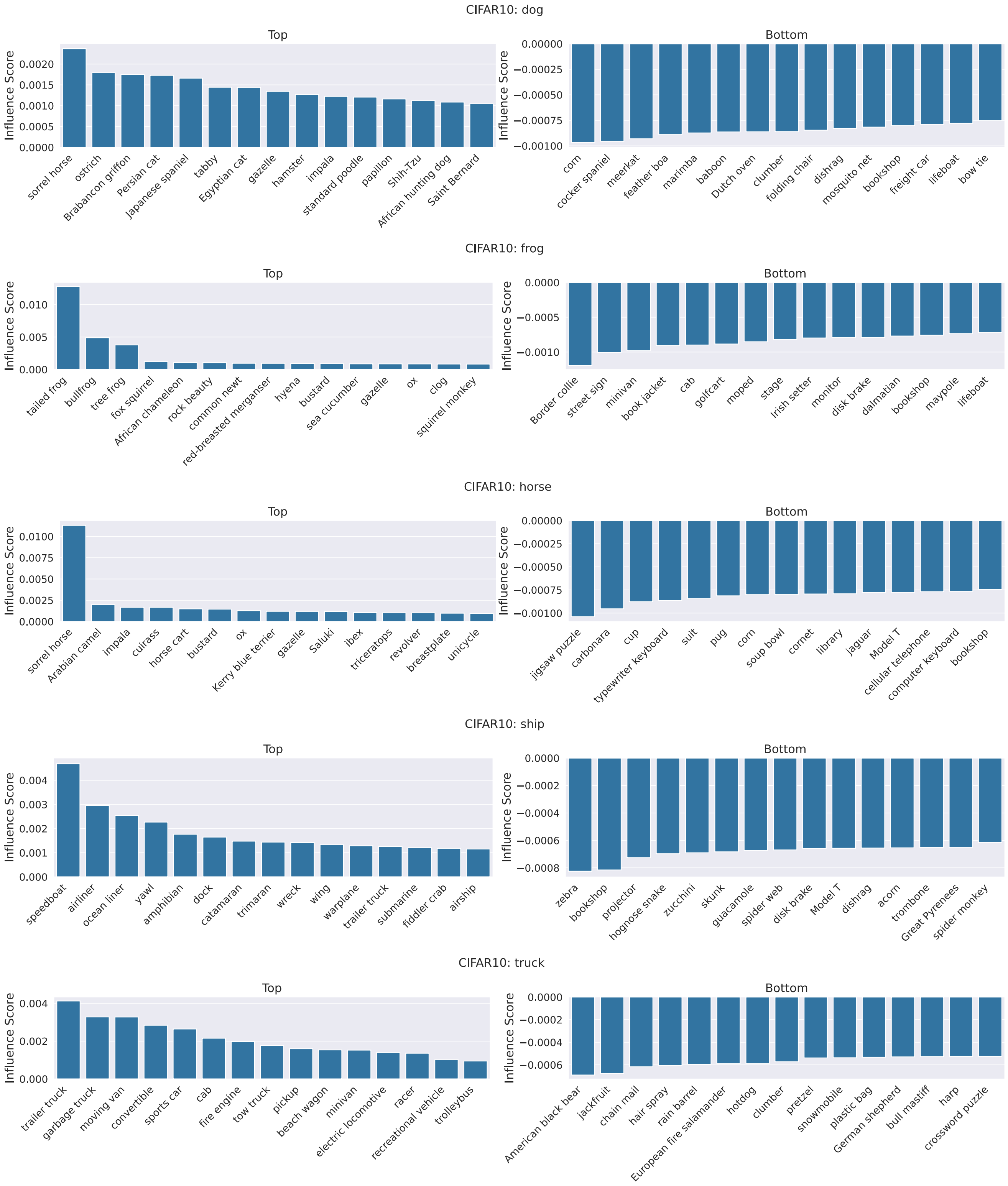
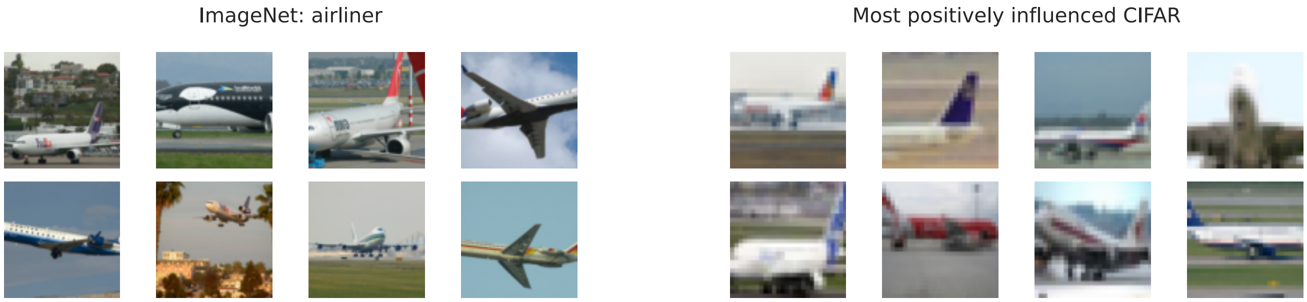


Figure 24. Top and bottom influencing ImageNet classes for all CIFAR-10 classes.

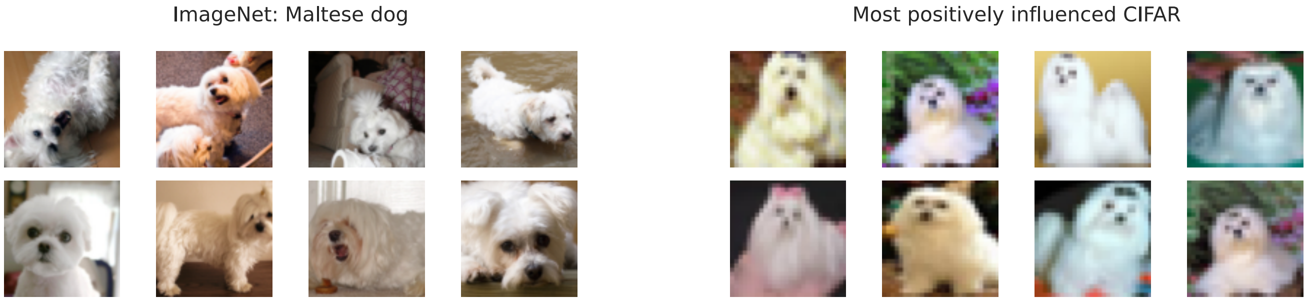
### E.2. More examples of extracted subpopulations from the target dataset

Here, we depict more examples of extracting subpopulations from the target dataset (as in Figure 4 of the main paper).


ImageNet: airliner

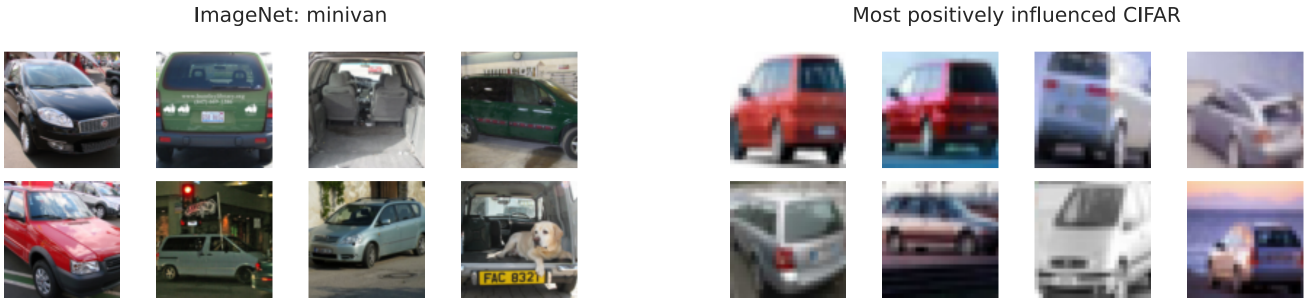
Most positively influenced CIFAR

(a) Airliner


ImageNet: Maltese dog

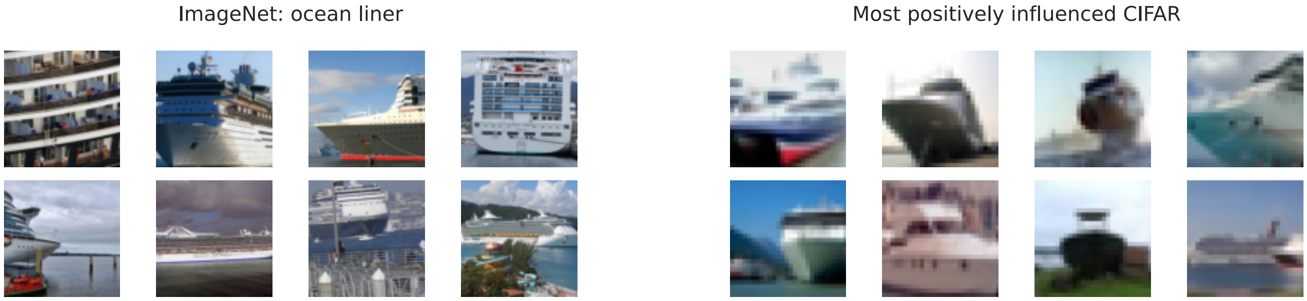
Most positively influenced CIFAR

(b) Maltese Dog


ImageNet: minivan

Most positively influenced CIFAR

(c) Japanese Spaniel


ImageNet: ocean liner

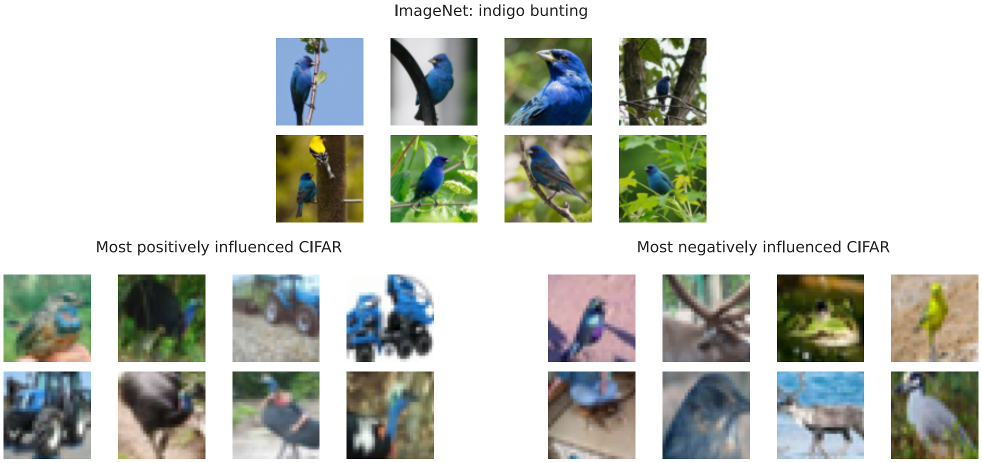
Most positively influenced CIFAR

(d) Ocean Liner

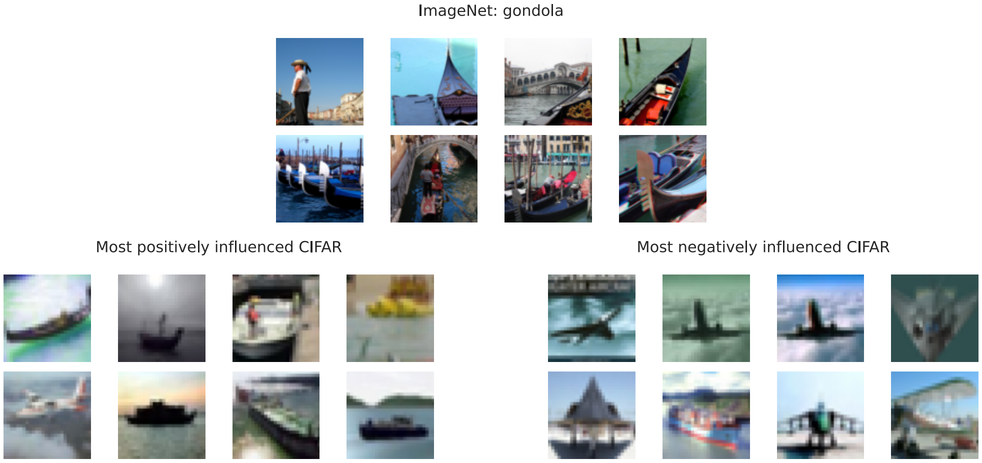
Figure 25. For each ImageNet class, we show the CIFAR-10 examples which were most positively influenced by that ImageNet class.

## E.3. More examples of transfer of shape and texture feature

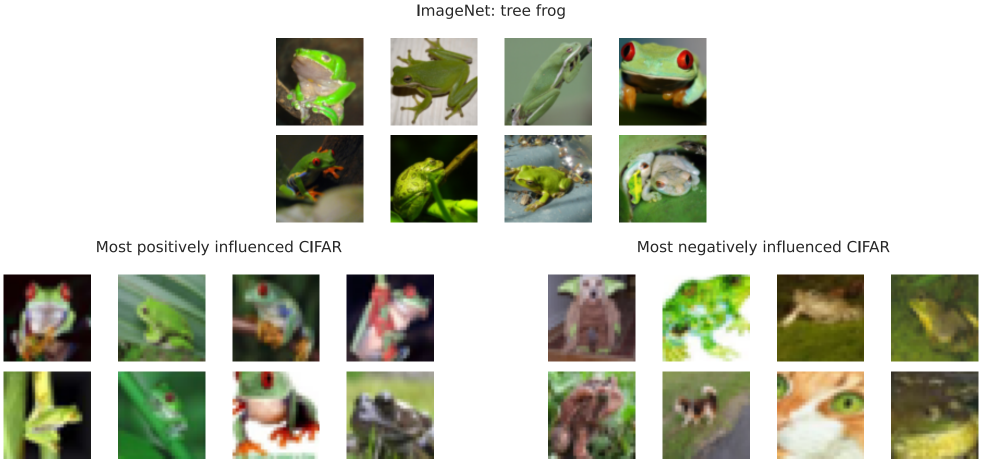
We depict more examples of ImageNet influencers which transfer shape or texture features (as in Figure 5).


ImageNet: indigo bunting
Most positively influenced CIFAR
Most negatively influenced CIFAR

(a) Indigo Bunting


ImageNet: gondola
Most positively influenced CIFAR
Most negatively influenced CIFAR

(b) Gondola

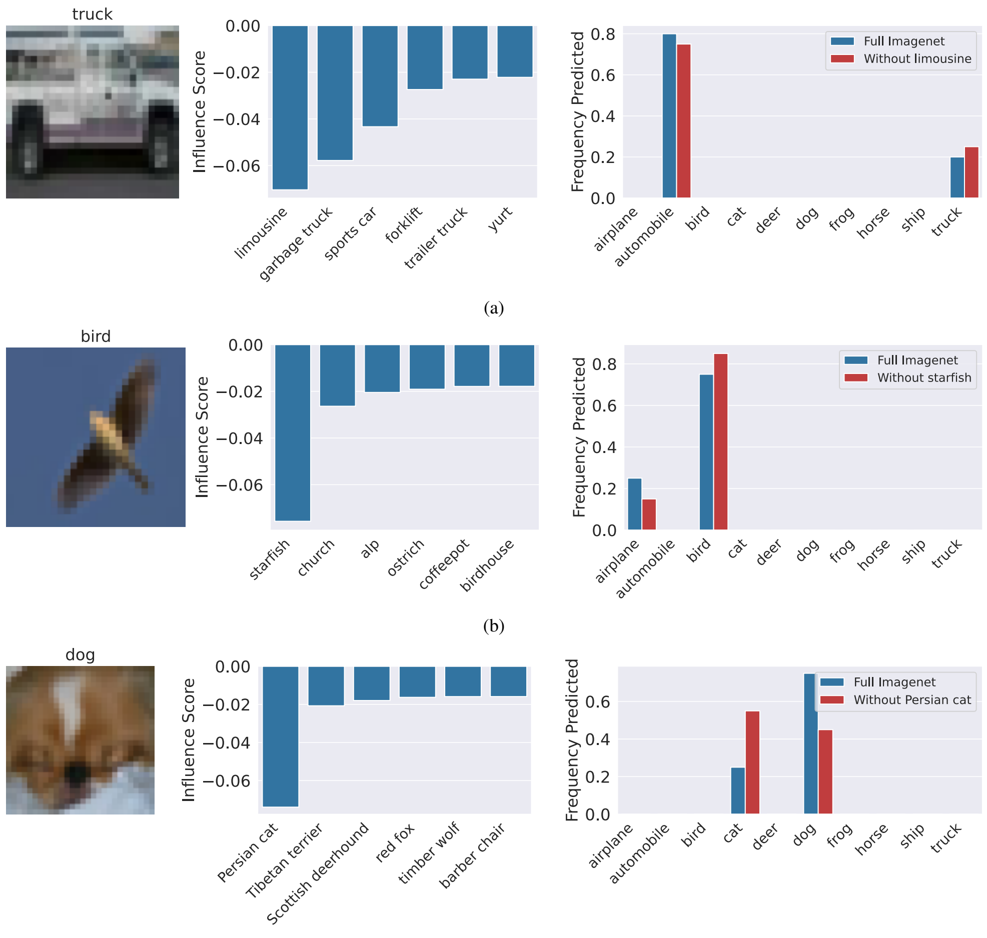

ImageNet: tree frog
Most positively influenced CIFAR
Most negatively influenced CIFAR

(c) Tree Frog

Figure 26. For each ImageNet class, we show the CIFAR-10 examples which were most highly influenced by that ImageNet class.

## E.4. More examples of debugging mistakes of transfer model using influencers

We display more examples of how our influences can be used to debug the mistakes of the transfer model, as presented in Figure 6 in the main paper. We find that, in most cases (see Figures 27a, 27b below and Figure 6 from the main paper), removing the top negative influencer improves the model's performance on the particular image. There are a few examples where removing the top negative influencer hurts the model's performance on the image (Figure 27c).



(a)

(b)

(c)

Figure 27. More examples of debugging transfer mistakes through our framework (c.f. Figure 6 in the main paper). For each CIFAR-10 image (**left**), we plot their most negative influencers (**middle**). On the **right**, we plot for each image the fraction (over 20 runs) of times that our transfer model predicts each class with and without the most negative influencer. While in most cases (a, b, c) we find that removing the most negative influencer helps the model predict the image correctly, in some cases removing the negative influencer does not outweigh the impact of removing images from the source dataset (d).

**Quantitative analysis.** How often does removing the most negative influencer actually improve the prediction on an image? For each of the following 14 classes, we run 20 runs of the ImageNet → CIFAR-10 fixed transfer pipeline while excluding that single class from the source dataset: ["sorrel horse", "limousine", "minivan", "fireboat", "ocean liner", "Arabian camel", "Persian cat", "ostrich", "gondola", "pool table", "starfish", "rapeseed", "tailed frog", "trailer truck"]. We compare against running the pipeline with 20 runs of the entire ImageNet dataset. Then, we look at individual CIFAR-10 images which were highly negatively influenced by one of these ImageNet classes, and check whether the images were predicted correctly more or less often when the top negative influencers were removed from the source dataset.

Of the 30 most negatively influenced ImageNet class/CIFAR-10 image pairs, 26 of them had the most negative ImageNet influencer in the above classes. Of those, 61.5% were predicted correctly more often when the negatively influential ImageNet class was removed, 34.6% were predicted incorrectly more often, and 3.9% were predicted correctly the same number of times.

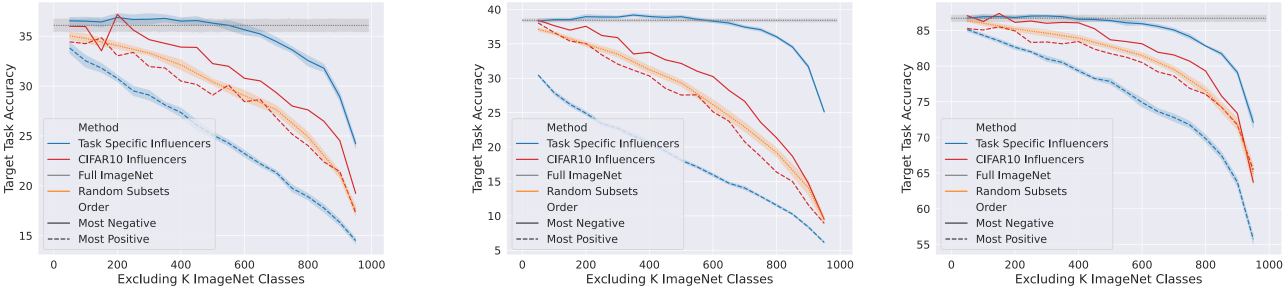
We then examine the top 8 most influenced CIFAR-10 images for each of the above 14 ImageNet classes. Of those 112 images, 53% were predicted correctly more often when the image was removed, 34% were predicted incorrectly more often, and 14% were predicted correctly the same number of times.

We thus find that, for the most influenced CIFAR-10 images, removing the top negative influencer usually improves that specific image's prediction (even though we are removing training data from the source dataset).

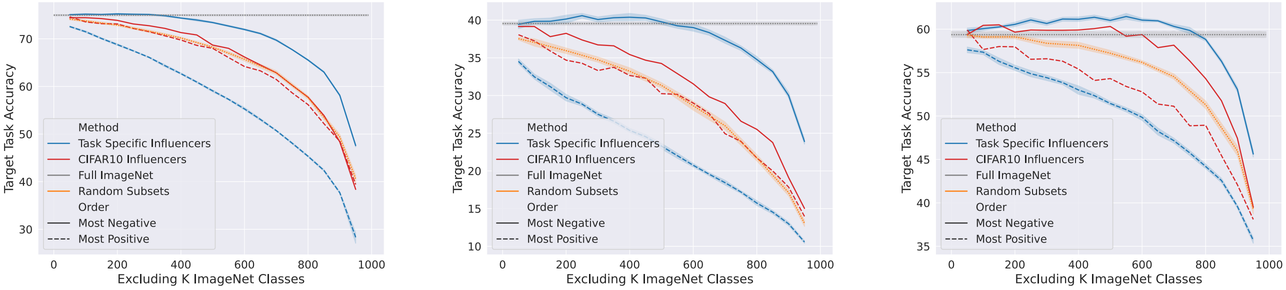
## E.5. Do Influences Transfer?

### E.5.1 Transfer across datasets

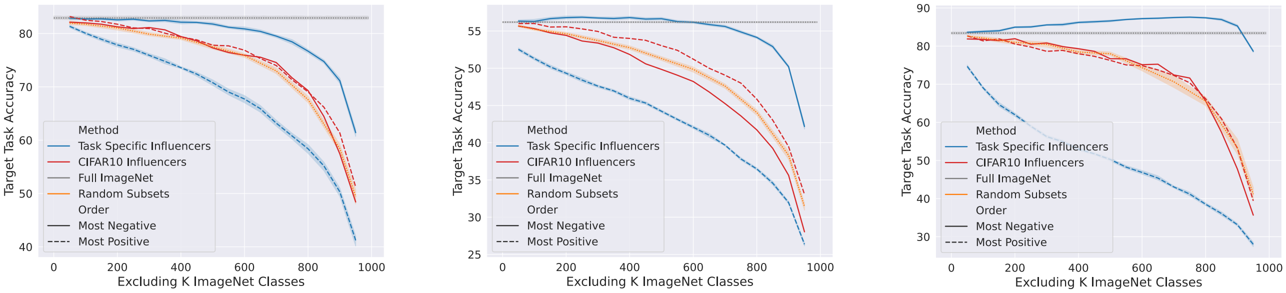
In this section, we seek to understand how much task-specific information is in the transfer influences that we compute. To do so, we use the transfer influences computed for CIFAR-10 in order to perform the counterfactual experiments for other datasets. We find that while using the CIFAR-10 influence values for other target datasets is more meaningful than random, they do not provide the same boost in performance when removing bottom influencers as using the task-specific influences. We thus conclude that the influence values computed by our framework are relatively task-specific.



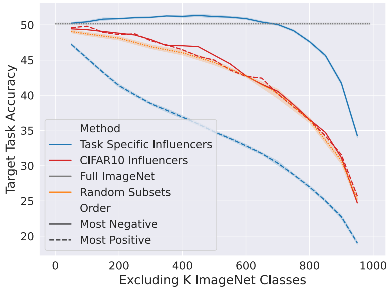
(a) AIRCRAFT (b) BIRDSNAP (c) CALTECH101



(d) CALTECH2556 (e) CARS (f) CIFAR100



(g) FLOWERS (h) FOOD (i) PETS



(j) SUN397

Figure 28. We repeat the counterfactual experiments (c.f. Figure 2b from the main paper) but using the influence values computed for CIFAR-10 on other target datasets.

### E.5.2 Transfer across architectures

How well do our transfer influences work across architectures? Recall that we computed our transfer influences using a ResNet-18. We now repeat the counterfactual experiment from Figure 2b, using the ResNet-18 influences to remove classes from the source dataset when training a ResNet-50. We find that these influences transfer relatively well: we can thus use a smaller architecture when computing transfer influences in lieu of a larger model.

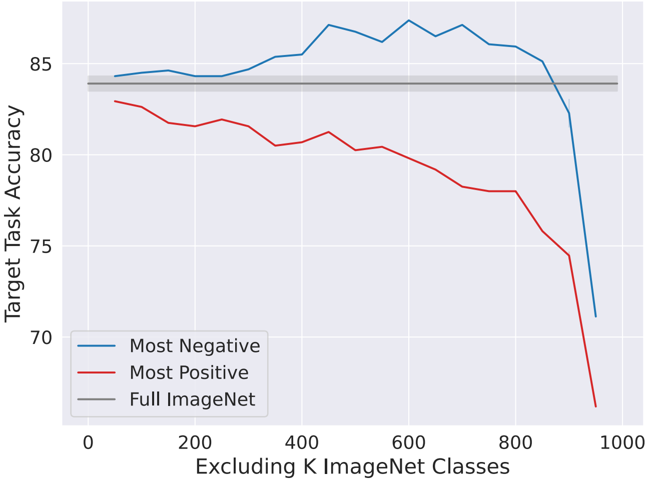
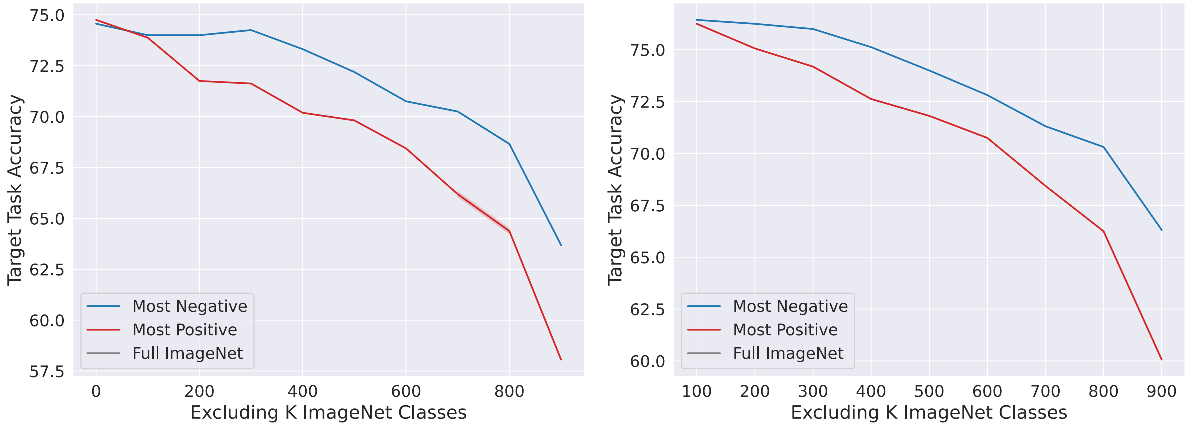


Figure 29. We repeat the counterfactual experiments (c.f. Figure 2b from the main paper) but use our influence values computed using a ResNet-18 on a ResNet-50.

We now perform the same experiment with vision transformers (ViT) [10], which use self-attention instead of convolutions (and thus behave differently than standard CNNs [20, 33]). Specifically, we evaluate our transfer influences on ViT-T and ViT-S (otherwise using the same hyperparameters as the ResNet-18). The ResNet-18 influences are still meaningful for the vision transformers, as removing the top classes degrades accuracy far more than the bottom classes. However, the ResNet-18 influences do not generalize as well to the ViTs as they did for the ResNet-50, as expected. In this setting, it may be more effective to use a small vision transformer (e.g., ViT-T) when computing the transfer influences.



ViT-T

ViT-S

Figure 30. We repeat the counterfactual experiments (c.f. Figure 2b from the main paper) but use our influence values computed using a ResNet-18 on a ViT-T and ViT-S.

## F. Further Convergence Analysis

In this section, we analyze the sample complexity of our influence estimation in more detail. In particular, our goal is to understand how the quality of influence estimates improves with the number of models trained. We also look at the impact of different choices of model outputs.

Instead of directly measuring our downstream objective (transfer accuracy on target dataset after removing the most influential classes), which is expensive, we design two proxy metrics to gauge the convergence of our estimates:

**Rank correlation.** As [19] shows, we can associate influences with a particular linear regression problem: given features $\mathbb{1}_{\mathcal{S}_i}$, an indicator vector of the subset of classes in the source dataset, predict the labels $f(t; \mathcal{S}_i)$, the model's output after it is finetuned on target dataset $\mathcal{T}$. In fact, we can interpret influences as weights corresponding to these binary features for presence of each class. That is, the influence vector $w_t = \{\text{Infl}[C_i \to t]\}_i$ defines a linear function that, given a subset of classes that the source model is trained on, predicts the corresponding model's output when trained on that subset. Here, we focus on analyzing average model accuracy, so in fact we consider the aggregate output $\sum_i f(t; \mathcal{S}_i)$ and the corresponding aggregated influences $w = \{\text{Infl}[C_i]\}_i$.

Given this view, we can measure the quality of the influence estimates by measuring their performance on the above regression problem on a held-out[9] set of examples $\{\mathcal{S}_i, \sum_i f(t; \mathcal{S}_i)\}$. In order to make different choices of model outputs (logit, confidence, etc.) comparable, we measure performance with spearman rank correlation between the ground truth model outputs and the predictions of the linear model (whose weights are given by the influences).

We measure this correlation while varying both the number of trained models used in the influence estimation and the choice of the model output, and the results are shown in Figure 31.

**False discovery rate.** Above we considered a measure based on predictive performance. Here, we focus on a more parameter-centric notion of False Discovery Rate (FDR). Intuitively, FDR here quantifies the following: how often are the top influencers actually just due to noise?

The Knockoffs [4] framework allows one to perform feature selection and also estimate the FDR. At a high level, it consists of two steps: First, one constructs "knockoff" versions of the original features which are distributed "indistinguishably" (more formally, exchangable) from the original features, and at the same time are independent of the response by design. Second, one applies an estimation algorithm of choice (e.g., OLS or LASSO) to the augmented data consisting of both the original and the knockoff features. Then, the relative frequency at which a variable $X_i$ has higher statistical signal than its knockoff counterpart $\tilde{X}_i$ indicates how likely the algorithm chooses true features, and this can be used to estimate the FDR (intuitively, if $X_i$ is independent of the response $y$, $X_i$ is indistinguishable from its knockoff $\tilde{X}_i$ and both are equally likely to have higher score).

We adapt this framework here (particularly, the verion known as model-X knockoffs) as follows:

1. Sample an independent knockoff matrix $\tilde{X}$ consisting of 1,000 binary features from the same distribution as the original mask matrix $X$ (namely, each instance has 500 active features).[10]
2. Estimate influences for both original and knockoff features using the difference-in-means estimator.
3. Consider the top $k = 100$ features by positive influence, and count the proportion of features that are knockoff. This yields an estimate of FDR among the top 100 features. An FDR of 0.5 indicates chance-level detection.[11]

As with the previous metric, we measure the above FDR for each target dataset while varying the number of trained models and the target output type (Figure 32).

**Discussion.** We observe the following from the above analyses using our two statistics:

- There are significant gains (higher correlation and lower FDR) with increasing number of trained models.
- But neither metric appears to have plateaued with 6,000 models, so this indicates that we can improve the accuracy of our influence estimates with more trained models, which may in turn improve the max improvement in transfer accuracies (Section 3), among other results.

---

[9] We split the 7,540 models into a training set of 6,000 and a validation set of the remainder.

[10] Technically, this procedure is not exactly valid in the original FDR framework as $X_i$ and $\tilde{X}_i$ are exchangable due to depenencies in the features. Nonetheless, it is accurate up to some approximation.

[11] This is different from the usual manner of controlling the FDR, but we look at this alternative metric for simplicity.

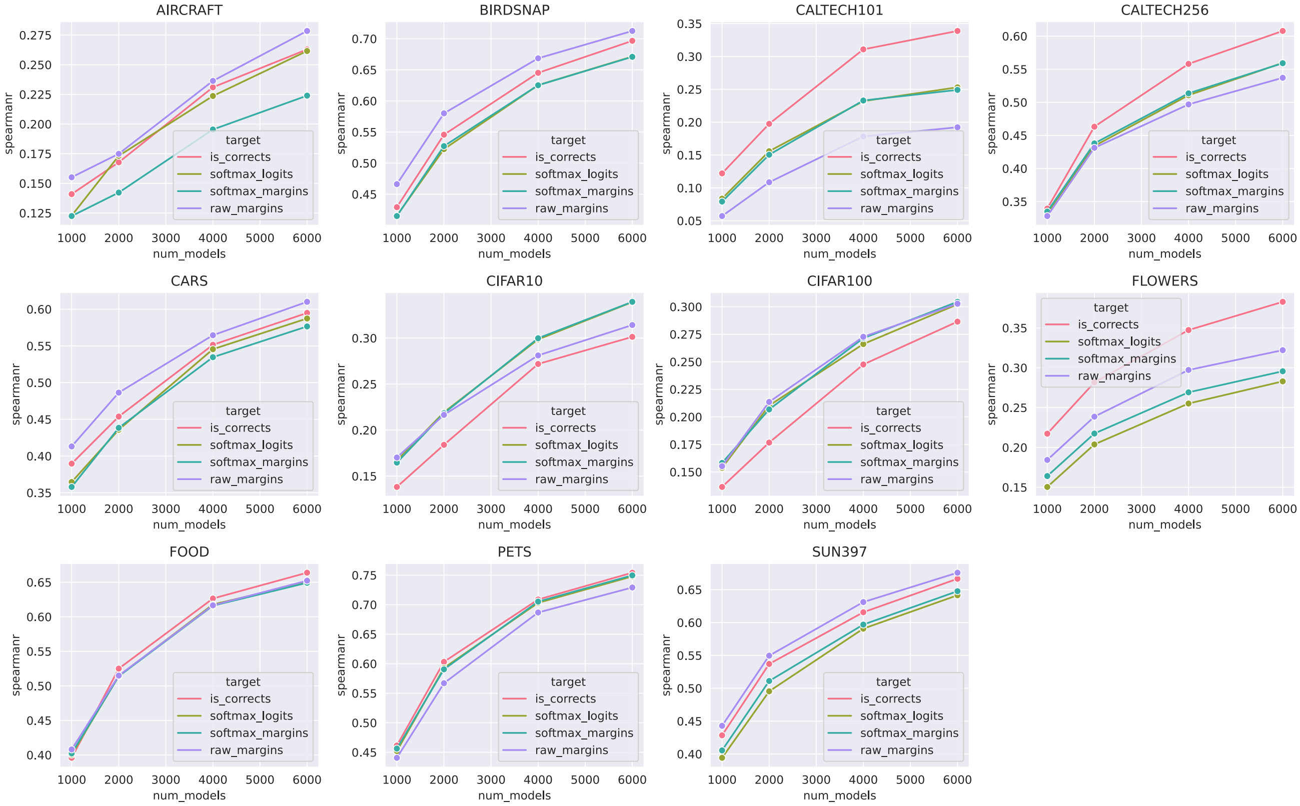


Figure 31. Measuring improvement in influence estimates using the **rank correlation** metric. The rank correlation here measures how well influence estimates perform in the underlying regression problem of predicting target accuracy from the subset of classes included in the training set. We evaluate on a held-out set of subsets independent from those used to estimate influences. Across all datasets, correlation improves significantly with more trained models.

- The choice of the target type does not appear to have a significant or consistent impact across different datasets, which is also consistent with the results of our counterfactual experiments (Appendix B.1).

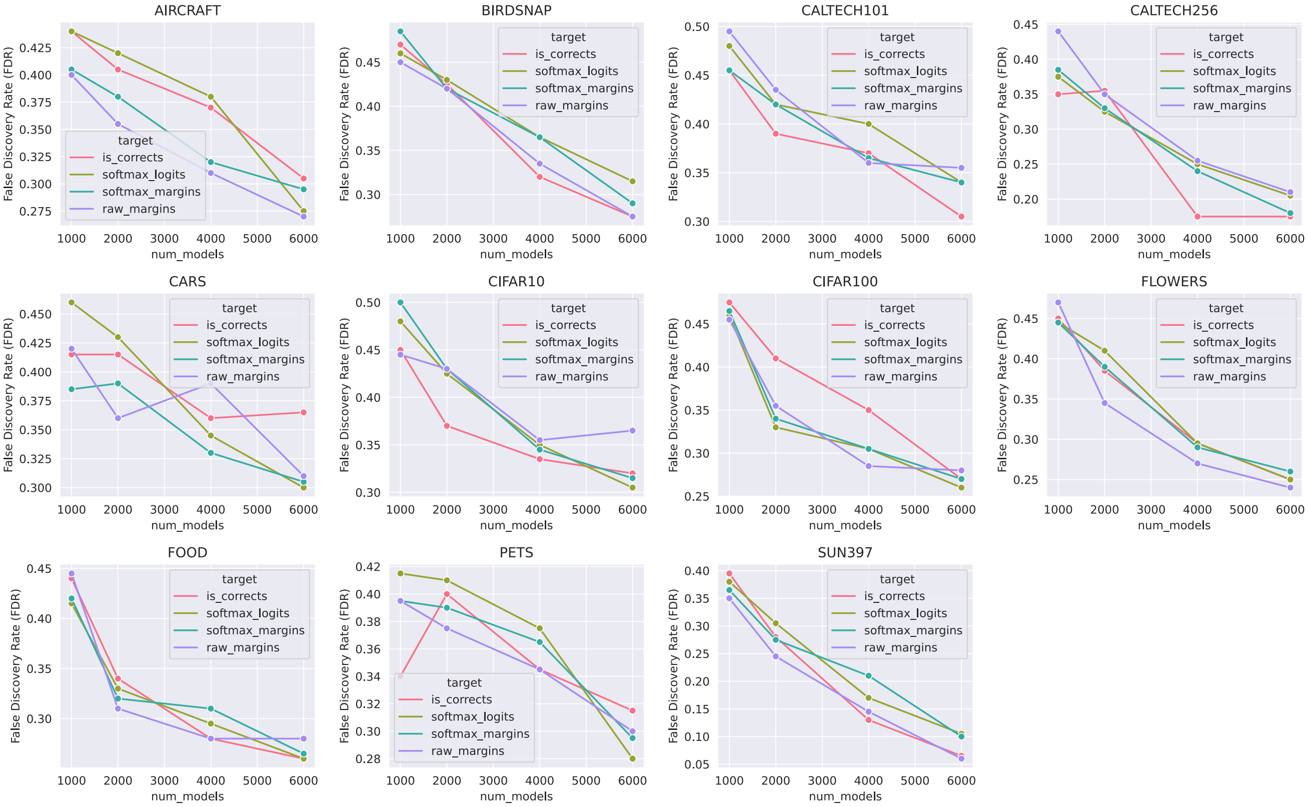


Figure 32. Measuring improvement in influence estimates using the **False Discovery Rate** heuristic. Using a procedure (loosely) based on the Knockoffs framework, we estimate the proportion of false discoveries within the top 100 features ranked by esteimated influences. Across all datasets, FDR decreases generally with more trained models.