Scalable Diverse Model Selection for Accessible Transfer Learning

Daniel Bolya[*]
Georgia Tech
dbolya@gatech.edu

Rohit Mittapalli[*]
Georgia Tech
rmittapalli3@gatech.edu

Judy Hoffman
Georgia Tech
judy@gatech.edu

Abstract

With the preponderance of pretrained deep learning models available off-the-shelf from model banks today, finding the best weights to fine-tune to your use-case can be a daunting task. Several methods have recently been proposed to find good models for transfer learning, but they either don't scale well to large model banks or don't perform well on the diversity of off-the-shelf models. Ideally the question we want to answer is, "given some data and a source model, can you quickly predict the model's accuracy after fine-tuning?" In this paper, we formalize this setting as "Scalable Diverse Model Selection" and propose several benchmarks for evaluating on this task. We find that existing model selection and transferability estimation methods perform poorly here and analyze why this is the case. We then introduce simple techniques to improve the performance and speed of these algorithms. Finally, we iterate on existing methods to create PARC, which outperforms all other methods on diverse model selection. We have released the benchmarks and method code[†] in hope to inspire future work in model selection for accessible transfer learning.

1 Introduction

Deep Neural Networks (DNNs) have shown to be very capable of solving a wide variety of visual tasks. However, these networks often require large amounts of data and training time to perform well, limiting the *accessibility* of deep learning for computer vision. One approach to alleviate this problem is to employ transfer learning, commonly by fine-tuning an off-the-shelf model on the desired task.

With the increasing number of off-the-shelf models available spanning different tasks, datasets, training methods, and architectures, choosing the best model from which to transfer is a challenging endeavor. A common heuristic in computer vision has been to use models pretrained on ImageNet [7] (specifically the ILSVRC challenge data), but more recent work is starting to expose weaknesses in the generalization performance of ImageNet features [39, 28, 40], and many of these works find that other pretraining datasets may perform better for some target tasks. Aside from source dataset, which architecture to select isn't clear (when comparing similarly fast models). One could theoretically train several transfers from a diverse set of pretrained weights and keep the best performing model, but this isn't feasible in most practical applications where compute is limited.

This motivates the need for a model selection method that could query a large bank of existing pretrained models (of which several already exist, e.g., [36, 56, 18]) with a small subset of the target data and return a set of weights which performs well when fine-tuned on all target data. Because of the sheer quantity of pretrained weights available to download off the internet today, such a model bank has the potential to be massive and cover a wide variety of tasks, datasets, and architectures. Any

[*]Equal Contribution
[†]https://dbolya.github.io/parc/

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

selection method that intends to operate on such a massive library of weights would thus need two important properties: it would need to be *scalable* in order to accommodate 100s to 1000s of source models, and it would need to perform well on a *diverse* set of input weights due to the wide variety of models available. In this paper, we formalize this setting as *Scalable Diverse Model Selection*.

Currently, there are two main lines of work that could address this scenario. The first is model selection [11, 46, 10], which attempts to select a viable transfer from a suite of arbitrary pretrained models. However, existing approaches require an initial model trained on the target data to suggest the transfer (which limits accessibility) and are not very successful when comparing across architectures (see Tab. 1). The second line of work is in transferability estimation [4, 52, 34], which attempts to predict how well a source model will transfer to a target task. While this might seem similar to model selection, the two tasks are subtly different in evaluation. Transferability estimation methods usually fix one source model and vary the target task (i.e., "For my source model, which task would it transfer to the best?"), while model selection methods do the opposite: fix a target task and vary the source models (i.e., "For my target task, which source model will transfer the best?"). While it might seem the same methods could work for either case, this turns out to not be the case (see Tab. 2).

Contributions. We formalize the task of *Scalable Diverse Model Selection*, which intends to make deep learning for computer vision more accessible. While other papers might have explored aspects of this space already, we standardize it by introducing several tools and benchmarks for evaluating model selection methods in this setting. First, we provide a controlled environment that includes exhaustively trained transfers from 8 source datasets to 6 target datasets across 4 different commonly used architectures for a total of 168 ground truth transfers for analysis (Sec. 3). We show that current state-of-the-art transferability and model selection methods fail to beat simple baselines in this new setting (Tab. 1). We then analyze why this is the case and provide techniques to improve performance (Sec. 4). Using insights from this analysis, we develop PARC, a method that outperforms other methods on this benchmark (Sec. 5). Finally, we show that these results generalize to a larger experiment with an extra dataset and 33 additional off-the-shelf pretrained models downloaded from the internet (for a total of 65 source models and 423 transfers) and by extending PARC to object detection (Sec. 6). We have released all benchmarks and evaluation code at `https://dbolya.github.io/parc/` in hopes to further the development of this promising area of research.

2 Related Work

In this paper, we introduce a new task for Scalable Diverse Model Selection, which attempts to make transfer learning more accessible by selecting the best pretrained model for a downstream task from a massive bank of off-the-shelf models. This is adjacent to several fields such as accessible transfer learning, transferability estimation, and Taskonomy model selection.

Transfer Learning. Transfer learning is the act of using a model trained in one setting to boost the performance or speed up the training of a model in another setting, and is an extensively studied field in deep learning [49]. Several approaches exist for transfer learning from a source model to a target dataset, the most popular of which include fine-tuning a new head to the target domain [9, 43, 60, 39], and fine-tuning the entire network [2, 17]. While there are more sophisticated methods for fine-tuning (e.g., [19, 37]), we study fine-tuning the entire network, as it's simple and widely adopted.

Accessible Transfer Leaning. A few works have tried to make transfer learning more accessible. Neural Data Server [57] allows users to augment their own data with similar data indexed from several massive image datasets. While helpful in a limited data environment, this requires the user to have additional compute in order to train with the extra images. Iterating on this idea, Scalable Transfer Learning with Expert Models [38] suggests pretrained models to the user instead, relaxing the compute requirement. However, the focus of the paper is primarily on creating these pretrained "expert" models and not in selecting between them. We argue that there are several "expert" models already widely available and trained with a large amount of data (e.g., [47, 33, 39]), and thus focus on the model selection aspect of this problem. Both of these works use simple baselines to guide their model selection: Neural Data Server [57] uses the accuracy of a logistic classifier on the source features fit to the target task, while Scalable Transfer Learning [38] uses nearest neighbors with hold-one-out cross-validation. We benchmark both baseline techniques in our setting.

Transferability Estimation. There are several works that attempt to predict how well a model will transfer to new tasks, ranging from methods that attempt to assess a models capacity for transfer [23, 28, 40] to those that attempt to predict transfer accuracy [12, 3, 45]. Others attempt to predict the gap in generalization between training and test time [24, 53, 25]. There has also been a recent line of work that directly attempts to estimate transfer learning accuracy given a source model and target dataset [4, 52, 34]. This line of work is the most applicable to our setting because the only assumption they make on the source model is that it was trained on classification. While not ideal, this allows us to benchmark these methods (H-Score [4], NCE [52], and LEEP [34]) in our setting. Otherwise, Deshpande et al. [8] propose perhaps what is most directly applicable to our work. Their setting is very similar (and fairly concurrent), though not as diverse and with no restrictions on evaluation speed. In addition, LogMe [58] is concurrent work that also focuses on practical transferability estimation, but they don't release their benchmark and their setting is more narrow.

Taskonomy Model Selection. The Taskonomy [59] models and dataset has been used to benchmark a previous line of model selection algorithms [11, 46, 10]. Taskonomy attempts to model similarities between tasks by how well models transferring from one task to another perform after fine-tuning. This makes it a natural test-bed for model selection evaluation, as it contains a large number of pretrained transfers to use as a benchmark. However, using Taskonomy as a benchmark is incomplete. All Taskonomy models are trained on the same data (with different labels) and follow roughly the same architecture, only varying the source and target task. This means that the benchmark doesn't test robustness to source model *diversity*, which is a core tenant of this work. Furthermore, typical model selection works on Taskonomy require a network trained on the target data to suggest transfers, which should be avoided to make transfer learning more accessible. We benchmark the performance of RSA [11] and DDS [10], as they are the best performing methods on Taskonomy model selection. There are other works in this space such as [6] and [1], but the former scales poorly with the number of sources, and the latter cannot compare across architectures by design, so we don't include either in our experiments.

3 Scalable Diverse Model Selection

In this work, we address the problem of model selection for transfer learning but through an expressly practical lens. The goal of model selection from a practitioner's point of view is to find a pretrained off-the-shelf model that will perform well after fine-tuning on their data. In order for a model selection method to perform well in this setting, the models it selects from need to be *diverse* (i.e., cover a wide variety of source datasets, architectures, and pretraining tasks), and the selection method needs to be *scalable* (since the number of off-the-shelf models available today is massive and growing).

In order to provide a realistic transfer learning model selection scenario, we propose to select a model from a *large* source model bank (over 100 models spanning several datasets and architectures) that transfers well to a target training set after full model fine-tuning. The naive approach to this problem would be to simply fully fine-tune a transfer from each source model to the target training set, but this would of course be computationally infeasible. We'd like to work in the practical setting where we don't train any extra models, so we'd like a *computationally efficient* transferability estimation method to predict which source models would transfer well. For the same reason, it's also infeasible to use the entire target training set for this estimate, since extracting all the features for 100 different source models is akin to training a new model for 100 epochs (though without the backward pass). Thus, we restrict the method to only use a small subset of the target training data for its estimate.

In this scenario, the target dataset is fixed while the source models can vary in dataset, architecture, and task. Previous work in model selection and transferability estimation typically only vary one of these aspects when evaluating their method (i.e., just task [59, 4, 11, 10], just dataset [34, 52], or just architecture [34], but not all three at once). It's unclear how well, if at all, these methods benchmarked by varying only one factor will perform when all of these factors can change. Thus, we've set out to test these methods in this much more challenging setting.

3.1 Creating a Diverse Benchmark

Because no model selection benchmark currently exists that varies more than one source factor at time, we create our own from 8 source datasets and 6 target datasets across 4 different architectures for classification, varying both source *dataset* and *architecture*. We will also vary task in Sec. 6.

Datasets and Architectures. An ideal selection of source datasets would contain related datasets, so that transfer learning makes sense. Thus, for this benchmark, we choose 6 well-known classification datasets of various difficulties that contain related subthemes: **Pets**: Stanford Dogs [26] and Oxford Pets [35], **Birds**: CUB200 [55] and NA Birds [54], and **Miscellaneous**: CIFAR10 [29] and Caltech101 [14]. We also include VOC2007 [13] and ImageNet 1k [7] as the 7th and 8th source datasets, but not as targets. The other 6 datasets are also included as targets.

When selecting a model, a practitioner is often interested in the accuracy-speed trade-off, meaning that the benchmark should include architectures at several tiers of evaluation speed. To facilitate this, we include three tiers of architectures: ResNet-50 [22] as the slowest, ResNet-18 [22] and GoogLeNet [48] in the middle, and AlexNet [30] as the fastest.

Evaluation. The goal in model selection is to find a source model that will transfer well to a target task. Ideally, this would be the best performing model, but that might be unreasonably difficult when there are 100s of models to choose from. Furthermore, practitioners are typically interested in the trade-off between performance and inference speed, which requires us to consider more than just the highest scoring models. What we really need in this case is a score for each model that correlates well with final fine-tuned accuracy on some target data. To benchmark such a score, we use Pearson Correlation [15], a widely adopted correlation metric (with 0 implying no correlation and 100 implying perfect correlation), between the model selection algorithm's transferability scores and the final fine-tuned accuracy.

Thus we employ the following procedure to test a model selection method $\mathcal{A}$ on our benchmark: for each target dataset $\mathcal{D}^t$ (indexed by $t \in T$), we sample an n image "probe set" $\mathcal{P}_n^t \subseteq \mathcal{D}^t$. Then, for each source model parameterized by θ_s (indexed by $s \in S$), we obtain

$$\alpha_s^t = \mathcal{A}(\theta_s, \mathcal{P}_n^t) \tag{1}$$

as the predicted score for how well the source model θ_s will transfer to the target dataset $\mathcal{D}^t$. Then, we train each transfer from θ_s to $\mathcal{D}^t$ and evaluate it on the test set of t to obtain the final transfer accuracies ω_s^t. Finally, we compute Pearson correlation (denoted by `pearsonr`) between the predicted and final transfer accuracy and average it over all target datasets:

$$\text{Mean PC (Varying Source)} = \frac{1}{|T|} \sum_{t \in T} \text{pearsonr}(\{\alpha_s^t : s \in S\}, \{\omega_s^t : s \in S\}) \tag{2}$$

Because there can be a large amount of variance in the probe set sampled, we further report mean and standard deviation over 5 different randomly sampled probe sets.

Note that we explicitly use Pearson Correlation because it incorporates all data points, if all you care about is selecting the most accurate model (without limits on speed or other model parameters), other metrics such as top-k accuracy may be more suitable. Thus, we include some extra metrics in the Appendix.

Implementation Details. For simplicity, images from all datasets are resized to 224×224. When constructing the probe sets, we ensure that there are at least 2 examples of each class (necessary for several methods) and randomly subsample classes if this results in more than $n = 500$ images. We train all source models and transfers using SGD with no weights frozen (i.e., full fine-tuning) and employ grid search to find optimal hyperparameters for each target dataset, architecture pair. Previous work in this space assume that each target model is trained with exactly the same hyperparameters. However, in a practical setting we expect some tuning to be done when training on the target dataset, so we have done the same. This makes the selection task more challenging but also more aligned with best case transfer outcomes. All models are trained on Titan Xp GPUs and all transferability methods are evaluated on the CPU. Note that times should be taken as lower bounds, since they're evaluated with expensive hardware. Many would-be practitioners don't have such hardware at their disposal.

3.2 Benchmarking Existing Work

We use this benchmark to evaluate several recent model selection and transferability estimation methods. We also include some typical baselines to contextualize the performance of these methods.

Probability-Based Methods. We test two very recent transferability estimation methods, NCE [52] and LEEP [34]. Both operate similarly: given a source model parameterized by θ that predicts

Method	Input	Training Time	Source Task Agnostic	Target Task Agnostic	Mean PC (% ↑)	Time (ms ↓)
NCE [52]	$p_\theta(z \mid x), y$				2.1 ± 0.7	3.3 ± 4.6
LEEP [34]	$p_\theta(z \mid x), y$				10.8 ± 0.1	3.4 ± 4.3
H-Score [4]	$f_\theta(x), y$		✓		-5.4 ± 4.9	5049.3 ± 7200.2
RSA AlexNet [11]	$f_\theta(x)$	1 Hour	✓	✓	-1.4 ± 0.6	93.7 ± 19.1
DDS AlexNet [10]	$f_\theta(x)$	1 Hour	✓	✓	1.7 ± 0.4	65.8 ± 17.9
RSA ResNet-50 [11]	$f_\theta(x)$	3 Hours	✓	✓	57.3 ± 0.4	102.7 ± 15.6
DDS ResNet-50 [10]	$f_\theta(x)$	3 Hours	✓	✓	56.1 ± 0.3	37.8 ± 10.7
Heuristic	N / A		✓	✓	51.0 ± 0.0	N / A
1-NN CV	$f_\theta(x), y$		✓		$\mathbf{60.8 \pm 1.2}$	42.1 ± 8.3
Logistic	$f_\theta(x), y$		✓		$\mathbf{61.9 \pm 1.4}$	716.2 ± 633.2

Table 1: **Existing Work Underperforms on Diverse Source Models.** Average Pearson correlation and time taken per transfer averaged over the 6 target datasets of our benchmark for several existing model selection and transferability estimation methods. Some methods use source model probabilities $p_\theta(z \mid x)$, some use latent features $f_\theta(x)$, and some additionally require target data labels y. RSA and DDS need an extra model trained on the target data, for which we report average training time. All non-baseline methods either have almost no correlation (●) or take exorbitant amounts of time (●).

probabilities $p_\theta(z \mid x)$ over the source classes z, input target images x and estimate a joint probability $\hat{p}(z, y)$ between the source classes z and the target classes y. Then $\hat{p}_\theta(y \mid x)$ can be estimated as

$$\hat{p}_\theta(y \mid x) = \sum_{z_i} \hat{p}(y \mid z_i)\, p_\theta(z_i \mid x) \tag{3}$$

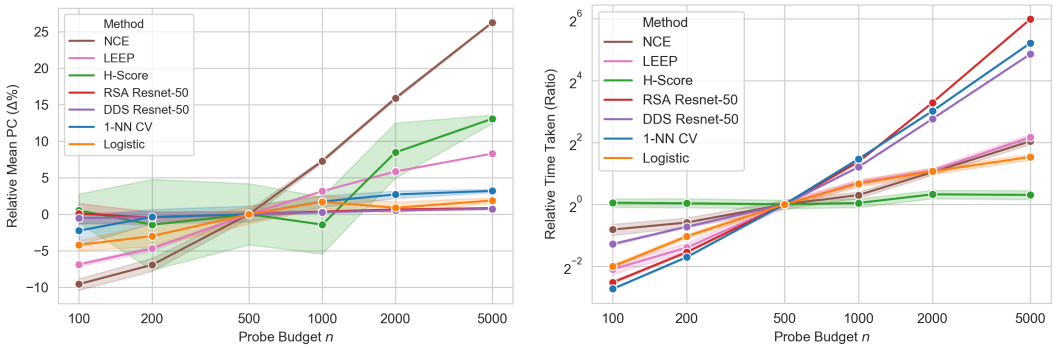
Transferability is then computed by aggregating this probability distribution for all target images x. NCE and LEEP differ mainly in how they produce $\hat{p}(z, y)$: NCE (as extended in [34]) produces this by counting when the source model predicts z for a target image with label y, while LEEP incorporates the entire distribution $p_\theta(z \mid x)$. The former naturally requires more data.

Feature-Based Methods. We also include several recent works from the Taskonomy [59] model selection literature: H-Score [4], RSA [11], and DDS [10]. All of these methods compare the penultimate layer's features $f_\theta(x)$ of a source model parameterized by θ evaluated on target images x to a different set of features. H-score compares the covariance of the features with the covariance of their mean over the target classes y. Note that H-Score scales poorly with latent feature size, since they invert the covariance of the features (which can be as big as 4096×4096 for AlexNet [30]).

RSA [11] and DDS [10], on the other hand, compare the source model's extracted features to that of a "probe" model already trained on the target data. They assume that if two images are far apart in the probe model's feature space, then they should also be far apart in the feature space of a good source model. RSA and DDS vary in how they construct these distances and how they aggregate them to produce final scores: RSA uses one minus the correlation coefficient between each pair of images for distance and Spearman correlation for the final score, while DDS tests a large number of combinations, of which we choose the best performing in [10] (cosine distance and z-score). Note that both of these methods require training an additional model before they can begin recommending transfers, and the architecture used for this probe model highly impacts performance (see Tab. 1). Because these methods requires extra training time and expert knowledge of what architecture to use, we include them only for reference. While 3 extra hours of training time on a Titan Xp might not seem like much, it drastically limits accessibility on cheaper hardware, where training can take days.

Baseline Methods. Finally, we include three fairly standard baselines. The first two are simple classifiers learned on top of the penultimate layer's features $f_\theta(x)$ and tested on the probe set to get an estimate of final fine-tuning accuracy (outputting accuracy as the score). For these methods, we include $k = 1$ nearest neighbors with leave one out cross-validation (denoted as k-NN CV, used in [38]) and a logistic classifier trained on one half of the probe set and evaluated on the other half (used in [57]). We also include a simple heuristic that rates performance as the number of layers in the source network ℓ_s plus the log of the total number of images in both the source and target sets:

$$\text{heuristic}_s^t = \ell_s + \log\left(|\mathcal{D}^s| + |\mathcal{D}^t|\right) \tag{4}$$



(a) Relative correlation to $n = 500$.

(b) Ratio of time taken to $n = 500$.

Figure 1: **Varying Probe Set Size.** We allow each model selection method to see a 500 image "probe set" of the target training data. To test whether this is a limiting factor for these methods, we vary probe set size and plot the resulting relative Pearson Correlation (a) and ratio of time taken (b) when compared to using 500 images. NCE [52], H-Score [4], and LEEP [34] all benefit significantly from more data (a), implying that their poor performance could be partially attributed to lack of data.

This captures the intuition that more layers and more data is better (with log data scaling).

Results. In Tab. 1 we list the Mean Pearson Correlation (computed as described in Sec. 3.1) and average time taken per transfer (not including feature extraction) for each method. We also include important properties of each method that can impact its scalability (training time, time taken) and whether it supports a diverse set of models (input requirements, being agnostic to source / target task).

From this experiment, we can see that current works fare poorly when applied to this new benchmark. In fact, none of these methods perform better than the simple $k = 1$ nearest neighbors baseline. Some even have close to no correlation (LEEP, NCE, H-Score). Furthermore, many of these methods are extremely expensive to compute (e.g., H-Score inverts a very large matrix when evaluating on AlexNet). RSA and DDS also require training of a target model with a manually selected architecture and high variability if poorly chosen (see Table 1). In the next section, we explore why these methods, which were not designed for our challenging transfer setting, may fail to generalize.

A Note on Scalability. This benchmark includes 168 transfers, which should be enough to test the scalability of model selection approaches. However, when evaluating current methods, we run the method on each source-target transfer and thus the "scalability" is linear per transfer. No current work exists to make this process sub-linear (e.g., through some kind of a weight embedding that you can binary search through), so instead we focus on the runtime speed of these methods. However, the end-goal of Scalable Diverse Model Selection is really to have sub-linear scaling, so we hope that future work can become even more scalable in that sense.

4 Analyzing Failure Modes of Existing Selection Methods

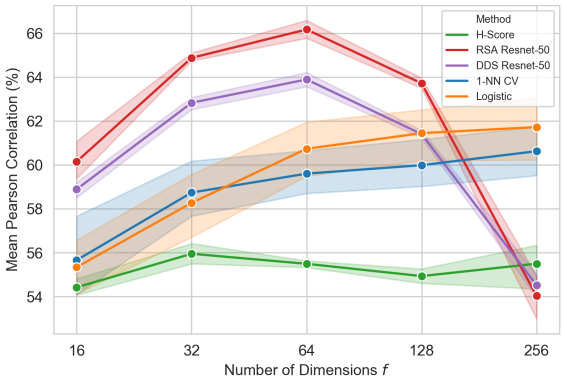
In this section, we explore what causes existing algorithms to fail in Sec. 3 and provide simple techniques to improve these methods on the challenging task of diverse model selection.

Small Probe Set Size. One potential reason why these methods fail to generalize to this setting is that, for efficiency, we use a probe set of $n = 500$ target train samples to compute transferability, while the final transfer model fine-tunes on all the data. In Fig. 1, we vary n and show relative correlation (a) and the ratio of time taken (b) for each method. NCE, LEEP, and H-Score all gain a significant boost from the extra data, implying that perhaps the restriction of a probe set is a major limiting factor for their performance. On the other hand, RSA, DDS, and the baseline methods seem to be able to perform well with even fewer samples, which is a boon to scalability.

Robustness to Evaluation Mode. There's a subtle, but important, point about evaluation that's often overlooked in transferability estimation. In model selection, you assume that you have a fixed

Method	**Source Dataset & Arch**	Architecture	Source Dataset	Target Dataset
NCE [52]	2.1 ± 0.7	21.1 ± 2.8	5.8 ± 0.4	78.5 ± 0.1
LEEP [34]	10.8 ± 0.1	8.0 ± 0.6	20.7 ± 0.2	67.6 ± 0.3
H-Score [4]	-5.4 ± 4.9	-11.0 ± 8.2	3.1 ± 8.8	-51.7 ± 7.0
RSA Resnet-50 [11]	57.3 ± 0.4	$\mathbf{67.5 \pm 0.6}$	61.5 ± 0.3	30.6 ± 2.7
DDS Resnet-50 [10]	56.1 ± 0.3	67.1 ± 0.3	62.3 ± 0.3	28.5 ± 3.2
Heuristic	51.05 ± 0.0	61.27 ± 0.0	7.13 ± 0.0	13.63 ± 0.0
1-NN CV	$\mathbf{60.8 \pm 1.2}$	$\mathbf{67.4 \pm 2.0}$	$\mathbf{67.0 \pm 0.9}$	$\mathbf{79.0 \pm 1.9}$
Logistic	$\mathbf{61.9 \pm 1.4}$	$\mathbf{68.7 \pm 2.9}$	$\mathbf{68.3 \pm 1.5}$	$\mathbf{81.0 \pm 1.7}$

Table 2: **Varying the Evaluation Mode.** In Eq. 2, we compute correlation over all source datasets and architectures. Here, we try varying each factor individually to more closely match each method's original setup (marked as ●). Most methods are poorly calibrated for other evaluation modes.



(a) Varying the number of PCA components.

Method	Mean PC (%)	Time (ms)
H-Score [4]	-5.4 ± 4.9	5049.3 ± 7200.2
HScore $f = 32$	$\mathbf{55.9 \pm 0.6}$	$\mathbf{43.1 \pm 21.4}$
RSA R-50 [11]	57.3 ± 0.4	$\mathbf{102.7 \pm 15.6}$
RSA R-50 $f = 64$	$\mathbf{66.2 \pm 0.5}$	183.4 ± 42.4
DDS R-50 [10]	56.1 ± 0.3	$\mathbf{37.8 \pm 10.7}$
DDS R-50 $f = 64$	$\mathbf{63.9 \pm 0.4}$	107.6 ± 32.1
1-NN CV	$\mathbf{60.8 \pm 1.2}$	$\mathbf{42.1 \pm 8.3}$
1-NN CV $f = 256$	60.6 ± 1.3	192.8 ± 40.9
Logistic	$\mathbf{61.9 \pm 1.4}$	716.2 ± 633.2
Logistic $f = 256$	61.7 ± 1.6	$\mathbf{339.4 \pm 155.9}$

(b) PCA can improve performance and speed.

Figure 2: **Calibrating Features with PCA.** Feature-based methods with PCA applied to the features beforehand for different numbers of outputs dimensions f (a). Applying feature reduction significantly improves all non-baseline methods (b), indicating that feature calibration was an issue. H-Score and Logistic, two methods that depend heavily on the number of features, also become much faster.

target dataset and would like to select the best source model for transfer. Transferability estimation often answers the opposite question: given a fixed source model, which target dataset will it best transfer to? The former evaluates following Eq. 2, while the latter considers the following correlation:

$$\text{Mean PC (Varying Target)} = \frac{1}{|S|} \sum_{s \in S} \text{pearsonr}(\{\alpha_s^t : t \in T\}, \{\omega_s^t : t \in T\}) \tag{5}$$

One might assume that the same method would work well in both situations, but this is not the case.

In Tab. 2, we test different evaluation settings by swapping what set correlation is being computed over (i.e., S for source and architecture, T for target, and subsets of S for the rest). For each non-baseline method, we've marked their native evaluation mode (using source dataset for Taskonomy). Most methods work well using their original evaluation protocol, but are not robust to other settings. Since each evaluation mode compares outputs across different factors, we hypothesize that the poor performance may be due to miscalibration and propose ways to mitigate this in the following sections.

Dimensionality Reduction. For the feature based methods, there's a lot of potential variation in the number of features between architectures. For instance, the number of features $|f_\theta(x)|$ for AlexNet is 4096, while for ResNet-18, it's 1024. Not only does this cause some methods to vary wildly in evaluation time (e.g., H-Score), but it also can be a source of miscalibration. To address this, we apply PCA dimensionality reduction [16] to the features, $f_\theta(x)$, down to a fixed dimension f before computing each selection method. The results for this experiment are displayed in Fig. 2. All non-baseline feature-based methods receive a significant boost in performance from this dimensionality

Method	Mean PC (%) Original	Mean PC (%) With Heuristic
LEEP [34]	10.8 ± 0.1	$\mathbf{26.8 \pm 0.1}$
NCE [52]	2.1 ± 0.7	$\mathbf{28.3 \pm 0.5}$
H-Score [4]	-5.4 ± 4.9	$\mathbf{25.6 \pm 9.2}$
RSA R-50 [11]	56.1 ± 0.3	$\mathbf{65.7 \pm 0.2}$
DDS R-50 [10]	57.3 ± 0.4	$\mathbf{64.9 \pm 0.2}$
1-NN CV	60.8 ± 1.2	$\mathbf{68.0 \pm 0.7}$
Logistic	61.9 ± 1.4	$\mathbf{69.2 \pm 1.1}$

Table 3: **Modeling Capacity to Change.** In this experiment, we add the depth ℓ_s of the source model to each transferability prediction. This results in a significant performance boost for all methods on our benchmark.

Method	Mean PC (%)	Time (ms)
1-NN CV $+\ell$	68.0 ± 0.7	42.1 ± 8.3
Logistic $+\ell$	69.2 ± 1.1	716.2 ± 633.2
RSA R-50 [11]	57.3 ± 0.4	102.7 ± 15.6
RSA R-50 $f = 32$	64.9 ± 0.2	183.4 ± 42.4
RSA R-50 $+\ell, f = 32$	68.2 ± 0.0	183.4 ± 42.4
PARC	53.0 ± 0.9	49.4 ± 18.3
PARC $f = 32$	59.3 ± 0.7	107.0 ± 31.1
PARC $+\ell, f = 32$	$\mathbf{70.3 \pm 0.5}$	107.0 ± 31.1

Table 4: **PARC Results.** PARC compared with all the beneficial tweaks in Sec. 4 (f for feature reduction and $+\ell$ for the layer heuristic, where $f = 32$ is best when combined). PARC outperforms all other methods, but requires these tweaks to work well.

reduction step. RSA and DDS even outperform the baselines after dimensionality reduction. For subsequent experiments, we will use the best value for f found in Fig. 2a for each method.

Capacity to Change. The goal of a scalable diverse model selection algorithm is predict a score that correlates well with full fine-tuning on the target set. However, all methods so far have used the source model as fixed features or probabilities, without considering the potential for those features to change after fine-tuning. Thus, we observe poor performance on more difficult target datasets (such as NA Birds, see the Appendix), where models with a high capacity of learning are required. While modeling the capacity for a network to learn is out of the scope of this work, we can substitute that with a simple heuristic. A strong indicator of how much information a CNN can learn that applies to most off-the-shelf architectures (e.g., [30, 44, 48, 22, 50]) is simply the depth of the model (i.e., number of layers). In Tab. 3, we compare the performance on our benchmark for each method before and after adding this heuristic. To integrate this intuition, we first normalize the predicted scores α_s^t for each method by their mean μ^t and standard deviation σ^t over all transfers and then add the relative source model depth ℓ_s over the maximum depth possible, ℓ_{max} (i.e., $\ell_{\text{max}} = 50$ for our experiments):

$$(\alpha_s^t)' = \frac{\alpha_s^t - \mu^t}{\sigma^t} + \frac{\ell_s}{\ell_{\text{max}}} \tag{6}$$

This change results in a significant boost in performance for all methods and will be denoted as $+\ell$ for future experiments. We explore other ways to incorporate this model capacity in the Appendix.

5 Pairwise Annotation Representation Comparison (PARC)

Following the insights gained in Sec. 4, we devise a new method for the task of scalable diverse model selection. RSA [11] with dimensionality reduction performs extremely well (see Fig. 2), however it requires a "probe model" trained on the target data. To alleviate this restriction, we perform an intuitive modification: we replace the probe model with the ground truth labels. Features that work well for a target task should consider two images dissimilar if they were annotated differently.

More formally, given a probe set $\mathcal{P}_n$ of target images x and labels y and a model parameterized by θ, PARC produces two distance matrices D_θ, D_y of shape $n \times n$ as

$$D_\theta = 1 - \text{corrcoef}(f_\theta(x)) \qquad D_y = 1 - \text{corrcoef}(g(y)) \tag{7}$$

where corrcoef computes pairwise Pearson product-moment correlation between the features of each pair of images (as used in RSA) and g maps the labels y to some vector representation. For classification, g maps y to the corresponding one-hot vector, but in general, g can be any function that maps the annotations to a vector. For instance, with semantic segmentation as the target task, g could produce a pixel-wise average of the annotations, and similar extensions exist for other computer vision tasks. We explore this further in Sec. 6.2.

Then, to compute the final PARC score, like RSA we simply compute the Spearman correlation between the two distance matrices for all pairs of images:

$$\text{PARC}(\theta, \mathcal{P}_n) = \text{spearmanr}(\{D_\theta[i,j] : i < j\}, \{D_y[i,j] : i < j\}) \tag{8}$$

Method	Mean PC (%)	Time (ms)
RSA R-50 $+\ell, f = 64$	50.64 ± 0.21	180.6 ± 29.3
DDS R-50 $+\ell, f = 64$	50.72 ± 0.19	120.4 ± 26.2
1-NN CV $+\ell, f = 256$	51.35 ± 0.67	209.6 ± 61.5
PARC $+\ell, f = 32$	$\mathbf{52.04 \pm 0.52}$	122.9 ± 29.1

Table 5: **Crowd-Sourced Models.** A more general benchmark obtained by training transfers from arbitrary crowd-sourced models. This includes 65 models from a variety of domains and 423 total transfers.

Method	Training Time	PC (%)
RSA F-RCNN	12 Hours	$\mathbf{96.33 \pm 0.31}$
DDS F-RCNN	12 Hours	95.97 ± 0.68
1-NN CV	**None**	89.67 ± 6.09
PARC	**None**	92.20 ± 5.63

Table 6: **Object Detection.** Predicting transfer performance for object detection. We employ a simple scheme to extend classification-based methods to object detection.

Results. We apply all the same improvements discussed in Sec. 4 and report the performance of PARC on our benchmark in Tab. 4. With both the dimensionality reduction and heuristic ensemble, PARC outperforms every other method (even with the same improvements). PARC observes a much larger boost from the heuristic ensemble than RSA, likely because RSA is allowed to use a ResNet-50 model fine-tuned on the target set. Thus, it already has some information about how features can change over fine-tuning built in (which is what the heuristic is trying to accomplish). PARC, on the other hand, isn't allowed this extra information, and thus adding the heuristic helps with calibration tremendously. We include more results for PARC in the Appendix.

6 Extended Benchmarks

Our benchmark in Sec. 3 describes a more practically useful setting than previous works, but it doesn't fully encapsulate scalable diverse model selection. Ideally, we could predict transferability between any source model (for any dataset, architecture, or task) to any target dataset or task. In this section, we explore arbitrary crowd-sourced source models and object detection as the target task.

6.1 A Crowd-Sourced Benchmark

In this experiment, we test how well PARC and other model selection methods perform on an even more diverse source model bank. To facilitate this, we collect 33 models from online model banks (i.e., [56, 18]) that span several source datasets ([7, 13, 32, 20, 51, 61, 33, 5]) and model architectures ([22, 42, 21, 27, 31]) covering many different pretext tasks (see the Appendix for full details). We combine this with the original 32 models we trained for a total of 65 source models and we add VOC2007 [13] multi-class classification as an additional target task, resulting in 423 transfers total. We place no restrictions on how the original models were trained, but we do normalize their features.

Results for this benchmark are available in Tab. 5. We only test on the subset of methods that support multi-class classification out of the box and apply all improvements from Sec. 4. While PARC outperforms the other methods, we note that none of the methods perform very well here (all having around 50% correlation with transfer accuracy). This indicates that selecting from an extremely diverse set of source models is still a very challenging task and warrants further study.

6.2 Transferability to Detection

So far, we've only considered classification as the target task. However, as mentioned in Sec. 5, we can apply PARC to other tasks by summarizing the annotations y with a vector $g(y)$, e.g., by averaging the annotations pixel-wise. To average "pixel-wise" for object detection, we count all pixels belonging to boxes for each class, and then normalize by the total area of all boxes in the image. To extend 1-NN, we measure the L_1 distance between pairs of these aggregate label vectors.

In Tab. 6, we display the performance of each method for predicting transfers from 6 source detectors (Faster and Mask R-CNN [42, 21] on Cityscapes [5] and COCO [32], and Retinanet [31] and YOLOv3 [41] on just COCO) transferring to VOC2007 detection [13]. We compute Pearson correlation between the predicted scores and the mAP of each fine-tuned model, and we provide RSA and DDS a Faster R-CNN model trained on VOC2007. Because there are only 6 transfers, this is a much easier experiment than our main benchmark (and thus we found the techniques discussed in Sec. 4 to not

be important). While it doesn't perform as well as RSA or DDS here, PARC requires no additional model trained on the target data. Yet, it is still highly correlated with final fine-tuned mAP.

7 Conclusion and Limitations

In this work, we introduce Scalable Diverse Model Selection, create several benchmarks to test this task, analyze existing methods in this setting, address multiple techniques to improve performance, and finally iterate on existing methods to create a new approach that works well. While the techniques we found combined with PARC improve performance on this setting, there are a few limitations of our work. First, we assume that the model selection method is applied to every source model, which could get extremely expensive. Subsequent work could try relaxing this assumption. Second, lowering the barrier to entry to transfer learning makes deep learning more accessible, but the model returned by these algorithms isn't guaranteed to be the best, so more work would likely need to be done to temper the expectations of non-experts. Finally, selecting from any arbitrary set of models in a scalable way still remains challenging (especially on crowd-sourced models) and thus is still an open problem. Many factors can affect fine-tuned performance, but we only consider source feature quality and architecture capacity in this paper. We address this latter point with a simple heuristic, which is slightly unsatisfying. For instance, an ideal system could have some comprehensive learnability score for each architecture instead. Several papers look at factors that affect fine-tuning performance in isolation, but none combine everything into one recommendation system. We hope that this paper can be the first step in creating such a diverse model selection algorithm. We believe that a robust system to recommend pretrained models for transfer learning would be an incredible boon for the accessibility of deep learning and hope that future work can study this task in further detail.

References

[1] Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless C Fowlkes, Stefano Soatto, and Pietro Perona. Task2vec: Task embedding for meta-learning. In *ICCV*, 2019. 3

[2] Pulkit Agrawal, Ross Girshick, and Jitendra Malik. Analyzing the performance of multilayer neural networks for object recognition. In *ECCV*, 2014. 2

[3] Haitham Bou Ammar, Eric Eaton, Matthew E. Taylor, Decibal C. Mocanu, Kurt Driessens, Gerhard Weiss, and Karl Tuyls. An automated measure of mdp similarity for transfer in reinforcement learning. In *AAAI Machine Learning for Interactive Systems Workshop*, 2014. 3

[4] Yajie Bao, Yang Li, Shao-Lun Huang, Lin Zhang, Lizhong Zheng, Amir Zamir, and Leonidas Guibas. An information-theoretic approach to transferability in task transfer learning. In *ICIP*, 2019. 2, 3, 5, 6, 7, 8

[5] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 9

[6] Yin Cui, Yang Song, Chen Sun, Andrew Howard, and Serge Belongie. Large scale fine-grained categorization and domain-specific transfer learning. In *CVPR*, 2018. 3

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 1, 4, 9

[8] Aditya Deshpande, Alessandro Achille, Avinash Ravichandran, Hao Li, Luca Zancato, Charless Fowlkes, Rahul Bhotika, Stefano Soatto, and Pietro Perona. A linearized framework and a new benchmark for model selection for fine-tuning. *Arxiv*, 2021. 3

[9] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, 2014. 2

[10] Kshitij Dwivedi, Jiahui Huang, Radoslaw Martin Cichy, and Gemma Roig. Duality diagram similarity: a generic framework for initialization selection in task transfer learning. In *ECCV*, 2020. 2, 3, 5, 7, 8

[11] Kshitij Dwivedi and Gemma Roig. Representation similarity analysis for efficient task taxonomy & transfer learning. In *CVPR*, 2019. 2, 3, 5, 7, 8

[12] Eric Eaton, Terran Lane, et al. Modeling transfer relationships between learning tasks for improved inductive transfer. In *ECML PKDD*, 2008. 3

[13] Mark Everingham, Luc Van Gool, Christopher Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010. 4, 9

[14] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *TPAMI*, 2006. 4

[15] David Freedman, Robert Pisani, and Roger Purves. Statistics (international student edition). *Pisani, R. Purves, 4th edn. WW Norton & Company, New York*, 2007. 4

[16] Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901. 7

[17] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 2

[18] Priya Goyal, Quentin Duval, Jeremy Reizenstein, Matthew Leavitt, Min Xu, Benjamin Lefaudeux, Mannat Singh, Vinicius Reis, Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Ishan Misra. Vissl. https://github.com/facebookresearch/vissl, 2021. 1, 9

[19] Yunhui Guo, Honghui Shi, Abhishek Kumar, Kristen Grauman, Tajana Rosing, and Rogerio Feris. Spottune: transfer learning through adaptive fine-tuning. In *CVPR*, 2019. 2

[20] Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *CVPR*, 2019. 9

[21] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 9

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 4, 8, 9

[23] Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. What makes imagenet good for transfer learning? *Arxiv*, 2016. 3

[24] Yiding Jiang, Dilip Krishnan, Hossein Mobahi, and Samy Bengio. Predicting the generalization gap in deep networks with margin distributions. *ICLR*, 2019. 3

[25] Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. *ICLR*, 2020. 3

[26] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Fei-Fei Li. Novel dataset for fine-grained image categorization: Stanford dogs. In *CVPR Workshop on Fine-Grained Visual Categorization (FGVC)*, 2011. 4

[27] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *CVPR*, 2019. 9

[28] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *CVPR*, 2019. 1, 3

[29] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 4

[30] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012. 4, 5, 8

[31] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017. 9

[32] 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 *ECCV*, 2014. 9

[33] 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 *ECCV*, 2018. 2, 9

[34] Cuong Nguyen, Tal Hassner, Matthias Seeger, and Cedric Archambeau. Leep: A new measure to evaluate transferability of learned representations. In *ICML*, 2020. 2, 3, 4, 5, 6, 7, 8

[35] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *CVPR*, 2012. 4

[36] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*. 2019. 1

[37] Matthew E Peters, Sebastian Ruder, and Noah A Smith. To tune or not to tune? adapting pretrained representations to diverse tasks. *Arxiv*, 2019. 2

[38] Joan Puigcerver, Carlos Riquelme, Basil Mustafa, Cedric Renggli, André Susano Pinto, Sylvain Gelly, Daniel Keysers, and Neil Houlsby. Scalable transfer learning with expert models. *Arxiv*, 2020. 2, 5

[39] 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. *Arxiv*, 2020. 1, 2

[40] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *ICML*, 2019. 1, 3

[41] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *Arxiv*, 2018. 9

[42] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *NeurIPS*, 2015. 9

[43] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *CVPR*, 2014. 2

[44] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *Arxiv*, 2014. 8

[45] Jivko Sinapov, Sanmit Narvekar, Matteo Leonetti, and Peter Stone. Learning inter-task transferability in the absence of target task samples. In *AAMAS*, 2015. 3

[46] Jie Song, Yixin Chen, Xinchao Wang, Chengchao Shen, and Mingli Song. Deep model transferability from attribution maps. *Arxiv*, 2019. 2, 3

[47] 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*, pages 843–852, 2017. 2

[48] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015. 4, 8

[49] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In *ICANN*, 2018. 2

[50] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019. 8

[51] Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. Yfcc100m: The new data in multimedia research. *Communications of the ACM*, 59(2):64–73, 2016. 9

[52] Anh T Tran, Cuong V Nguyen, and Tal Hassner. Transferability and hardness of supervised classification tasks. In *ICCV*, 2019. 2, 3, 4, 5, 6, 7, 8

[53] Thomas Unterthiner, Daniel Keysers, Sylvain Gelly, Olivier Bousquet, and Ilya Tolstikhin. Predicting neural network accuracy from weights. *Arxiv*, 2020. 3

[54] Grant Van Horn, Steve Branson, Ryan Farrell, Scott Haber, Jessie Barry, Panos Ipeirotis, Pietro Perona, and Serge Belongie. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In *CVPR*, 2015. 4

[55] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010. 4

[56] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 1, 9

[57] Xi Yan, David Acuna, and Sanja Fidler. Neural data server: A large-scale search engine for transfer learning data. In *CVPR*, 2020. 2, 5

[58] Kaichao You, Yong Liu, Jianmin Wang, and Mingsheng Long. Logme: Practical assessment of pre-trained models for transfer learning. In *ICML*, 2021. 3

[59] Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *CVPR*, 2018. 3, 5

[60] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. 2

[61] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. 2014. 9

A Per-Target Results

There was not enough room in the main paper to display the Pearson correlation of each method for each target dataset. We display full results for all methods here.

Note that there are two datasets that typically stand out as being more difficult than the others to predict transfer accuracy for: NA Birds and CIFAR-10, both for opposite reasons. CIFAR-10 is the easiest target dataset available by far, where any model can perform well after fine-tuning. This means that the source feature quality doesn't matter nearly as much. Since source feature quality is the only metric these methods use to predict transfer performance, they do poorly here. On the other hand, NA Birds is challenging and thus requires a high capacity model to transfer well. These methods also fail to take this factor into account.

A.1 No Tweaks

In Tab. 7, we display the Pearson Correlation for each target dataset individually. We also include results for additional baselines and skews of existing methods. k-NN (without CV) is nearest neighbors that, instead of hold-one-out cross validation, trains on half the probe set and tests on the other half (like Logistic). RSA and DDS Full are where the probe model provided to RSA or DDS is the same architecture as the current model being queried. This doesn't do nearly as well, perhaps because the probe features need to be consistent across transfers. We also provided RSA and DDS with GoogLeNet and ResNet-18 models here. Neither of those do as well as their ResNet-50 counterpart, but they perform better than AlexNet. This reiterates the point that the architecture used for the RSA or DDS probe model must be the best one available.

Method	Stan. Dogs	Ox. Pets	CUB 200	NA Birds	CIFAR 10	Caltech 101	Mean PC
LEEP	-6.3 ± 0.2	23.3 ± 0.6	25.8 ± 0.2	-5.9 ± 0.2	38.0 ± 0.6	-10.1 ± 0.3	10.8 ± 0.1
NCE	-8.8 ± 0.2	4.4 ± 0.3	13.3 ± 0.6	9.6 ± 1.3	15.3 ± 2.4	-21.2 ± 0.5	2.1 ± 0.7
HScore	-15.0 ± 16.9	-9.6 ± 12.7	9.5 ± 22.6	-3.8 ± 29.7	-5.3 ± 10.8	-8.4 ± 5.6	-5.4 ± 4.9
1-NN CV	75.2 ± 1.0	71.2 ± 1.7	53.6 ± 2.4	50.9 ± 4.2	54.1 ± 2.0	59.6 ± 0.9	60.8 ± 1.2
5-NN CV	75.0 ± 0.9	71.3 ± 1.7	53.0 ± 3.4	47.0 ± 9.1	51.7 ± 1.1	58.6 ± 0.9	59.4 ± 1.9
1-NN	74.4 ± 1.3	70.7 ± 2.7	52.6 ± 3.3	51.9 ± 5.0	52.5 ± 4.4	59.8 ± 1.6	60.3 ± 1.4
5-NN	75.4 ± 1.9	68.4 ± 2.3	58.1 ± 5.8	45.7 ± 10.9	47.6 ± 5.4	59.4 ± 2.3	59.1 ± 3.5
Logistic	75.2 ± 0.8	75.1 ± 2.3	55.6 ± 3.4	51.2 ± 2.4	51.5 ± 3.3	63.2 ± 1.6	61.9 ± 1.4
RSA Resnet-50	63.3 ± 0.6	75.9 ± 0.6	50.5 ± 1.5	41.5 ± 1.9	48.7 ± 1.5	63.7 ± 0.5	57.3 ± 0.4
RSA Resnet-18	63.0 ± 0.9	74.0 ± 0.7	37.7 ± 1.8	33.5 ± 2.3	50.8 ± 1.7	57.6 ± 0.9	52.8 ± 0.7
RSA GoogLeNet	47.8 ± 0.7	61.6 ± 0.5	5.9 ± 2.2	10.2 ± 3.1	31.5 ± 1.9	57.9 ± 1.6	35.8 ± 0.6
RSA Alexnet	-31.5 ± 1.6	27.9 ± 1.3	-17.0 ± 1.5	-45.6 ± 0.7	46.0 ± 0.3	11.8 ± 1.1	-1.4 ± 0.6
RSA Full	9.2 ± 0.7	63.0 ± 0.6	-32.1 ± 0.9	-67.8 ± 0.8	27.9 ± 1.8	31.0 ± 0.9	5.2 ± 0.5
DDS Resnet-50	62.4 ± 0.6	75.4 ± 0.5	49.8 ± 1.0	38.5 ± 0.9	49.1 ± 0.9	61.2 ± 0.5	56.1 ± 0.3
DDS Resnet-18	62.2 ± 0.8	74.2 ± 0.5	38.7 ± 1.2	31.8 ± 1.2	49.8 ± 1.5	57.4 ± 0.4	52.4 ± 0.4
DDS GoogLeNet	49.2 ± 0.8	64.6 ± 0.6	12.5 ± 1.4	12.6 ± 1.8	39.2 ± 1.1	57.7 ± 1.1	39.3 ± 0.5
DDS Alexnet	-27.8 ± 1.1	32.1 ± 1.3	-12.9 ± 1.5	-43.1 ± 0.7	47.2 ± 1.0	14.6 ± 0.9	1.7 ± 0.4
DDS Full	11.4 ± 0.5	63.5 ± 0.7	-27.0 ± 0.6	-63.3 ± 0.8	34.7 ± 1.6	30.9 ± 0.9	8.4 ± 0.4
PARC	58.9 ± 1.1	54.2 ± 2.9	45.0 ± 0.4	44.7 ± 2.9	47.0 ± 1.1	67.9 ± 1.2	53.0 ± 0.9

Table 7: **All Targets: No Tweaks.** Pearson Correlation for each target dataset without any tweaks applied for a budget size of $n = 500$.

A.2 Dimensionality Reduction

In Tab. 8, we show the Pearson Correlation for each dataset individually for each feature-based method while varying the dimensionality of the input features after PCA. The Mean PC column in this table is the data for the corresponding dimensionality reduction plot in the paper. We plot the means in Fig. 3.

Method	Stan. Dogs	Ox. Pets	CUB 200	NA Birds	CIFAR 10	Caltech 101	Mean PC
HScore $f = 16$	67.1 ± 1.0	68.2 ± 0.6	48.2 ± 1.3	32.4 ± 2.2	47.1 ± 1.7	63.6 ± 0.8	54.4 ± 0.4
HScore $f = 32$	69.9 ± 0.6	67.8 ± 0.8	50.5 ± 1.5	35.2 ± 0.8	48.0 ± 1.9	64.3 ± 0.5	55.9 ± 0.6
HScore $f = 64$	69.5 ± 0.8	69.2 ± 1.0	48.5 ± 1.4	31.0 ± 1.2	50.0 ± 0.8	64.8 ± 0.5	55.5 ± 0.2
HScore $f = 128$	68.3 ± 1.4	70.3 ± 1.0	46.8 ± 1.2	27.3 ± 2.6	50.5 ± 1.1	66.4 ± 0.6	54.9 ± 0.4
HScore $f = 256$	67.8 ± 0.9	71.2 ± 1.4	46.8 ± 1.5	30.0 ± 3.4	48.2 ± 0.9	69.0 ± 0.6	55.5 ± 1.2
RSA R-50 $f = 16$	68.9 ± 1.6	73.2 ± 0.6	56.8 ± 1.2	45.4 ± 3.5	55.5 ± 1.9	61.2 ± 1.4	60.2 ± 1.0
RSA R-50 $f = 32$	73.6 ± 0.8	71.7 ± 0.4	67.6 ± 1.1	52.6 ± 1.1	59.1 ± 0.9	64.8 ± 0.5	64.9 ± 0.2
RSA R-50 $f = 64$	71.9 ± 0.4	67.1 ± 0.6	73.2 ± 1.1	63.4 ± 2.3	58.3 ± 0.7	63.2 ± 0.6	66.2 ± 0.5
RSA R-50 $f = 128$	67.9 ± 0.5	61.2 ± 0.3	72.7 ± 2.1	65.1 ± 1.4	55.4 ± 0.5	60.0 ± 0.3	63.7 ± 0.3
RSA R-50 $f = 256$	59.9 ± 0.7	54.9 ± 0.6	63.2 ± 2.3	41.1 ± 3.2	48.0 ± 0.8	57.1 ± 0.7	54.0 ± 1.1
DDS R-50 $f = 16$	67.5 ± 1.0	71.7 ± 0.5	55.3 ± 0.7	42.7 ± 1.6	53.9 ± 0.6	62.2 ± 1.5	58.9 ± 0.4
DDS R-50 $f = 32$	71.1 ± 0.6	72.2 ± 0.5	63.0 ± 0.9	47.3 ± 1.4	58.0 ± 1.7	65.4 ± 0.5	62.8 ± 0.3
DDS R-50 $f = 64$	69.8 ± 0.4	71.2 ± 0.5	65.5 ± 1.2	51.9 ± 1.2	59.1 ± 1.0	65.8 ± 0.4	63.9 ± 0.4
DDS R-50 $f = 128$	66.4 ± 0.2	66.7 ± 0.3	64.8 ± 1.7	47.6 ± 0.4	57.4 ± 0.7	65.6 ± 0.5	61.4 ± 0.2
DDS R-50 $f = 256$	60.1 ± 0.5	58.9 ± 0.6	60.6 ± 1.3	34.7 ± 1.6	50.2 ± 0.3	62.7 ± 0.6	54.5 ± 0.6
1-NN CV $f = 16$	72.1 ± 1.8	70.1 ± 1.9	48.1 ± 2.2	35.3 ± 6.1	53.9 ± 3.6	54.6 ± 1.5	55.7 ± 2.1
1-NN CV $f = 32$	73.1 ± 0.9	71.0 ± 1.5	50.5 ± 2.1	45.9 ± 5.8	54.0 ± 2.2	58.0 ± 1.8	58.7 ± 1.5
1-NN CV $f = 64$	73.8 ± 1.3	72.0 ± 1.9	52.3 ± 2.6	46.8 ± 4.8	53.5 ± 2.5	59.2 ± 0.9	59.6 ± 1.2
1-NN CV $f = 128$	74.4 ± 1.1	71.7 ± 1.8	52.7 ± 2.1	48.2 ± 4.0	52.9 ± 1.2	59.9 ± 0.6	60.0 ± 1.2
1-NN CV $f = 256$	75.1 ± 1.2	71.1 ± 1.6	53.4 ± 2.4	50.1 ± 4.8	54.2 ± 1.6	59.9 ± 0.8	60.6 ± 1.3
Logistic $f = 16$	72.0 ± 1.7	71.3 ± 1.9	46.1 ± 3.1	36.8 ± 6.3	49.1 ± 6.0	56.8 ± 1.3	55.3 ± 1.5
Logistic $f = 32$	74.7 ± 0.8	72.2 ± 2.7	52.2 ± 3.0	43.0 ± 4.5	46.7 ± 4.9	60.8 ± 0.9	58.3 ± 1.6
Logistic $f = 64$	75.3 ± 1.2	73.8 ± 2.2	53.6 ± 2.9	48.8 ± 5.2	50.0 ± 4.3	62.9 ± 2.4	60.7 ± 1.4
Logistic $f = 128$	75.8 ± 1.1	74.1 ± 2.9	54.4 ± 2.6	49.2 ± 3.5	51.4 ± 1.7	63.7 ± 1.2	61.4 ± 1.3
Logistic $f = 256$	76.1 ± 0.7	74.4 ± 2.6	55.0 ± 2.8	50.5 ± 2.7	50.8 ± 2.8	63.6 ± 1.4	61.7 ± 1.6
PARC $f = 16$	64.3 ± 2.2	55.0 ± 7.7	44.7 ± 1.5	40.1 ± 1.8	49.9 ± 2.3	67.9 ± 1.1	53.6 ± 1.1
PARC $f = 32$	68.6 ± 0.6	68.2 ± 4.4	49.8 ± 3.0	48.8 ± 1.8	50.5 ± 1.9	70.0 ± 0.8	59.3 ± 0.7
PARC $f = 64$	71.4 ± 0.9	73.2 ± 3.7	53.3 ± 3.0	50.7 ± 4.2	50.1 ± 1.4	71.8 ± 1.4	61.7 ± 1.3
PARC $f = 128$	72.8 ± 1.5	72.3 ± 3.5	57.0 ± 2.1	50.9 ± 3.6	46.7 ± 2.0	72.5 ± 1.0	62.0 ± 0.7
PARC $f = 256$	75.3 ± 1.0	69.7 ± 3.9	60.1 ± 2.3	52.5 ± 4.0	38.8 ± 2.6	71.6 ± 0.8	61.3 ± 0.6

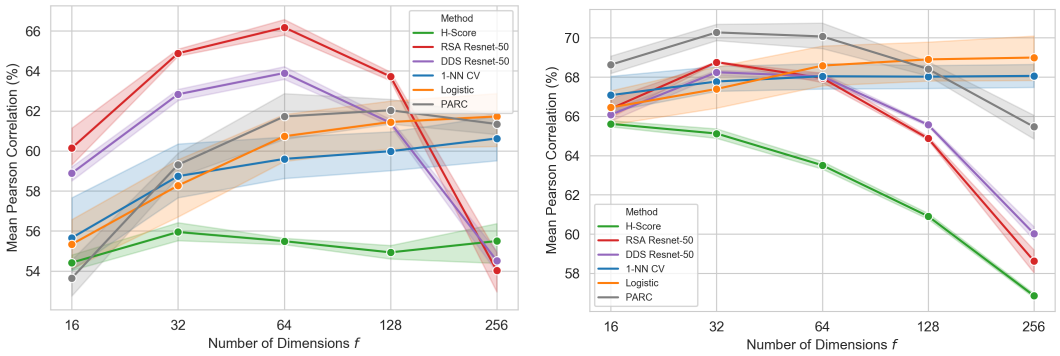
Table 8: **All Targets: Dimensionality Reduction.** Pearson Correlation for each target dataset with different levels of dimensionality reduction for a budget size of $n = 500$.

A.3 Capacity to Change

In Tab. 9, we include the Pearson Correlation for each dataset individually for all methods while incorporating the number of layers ℓ_s heuristic. We use this as a heuristic to gauge how well each source architecture can learn from complex data (with more layers predicting better performance). This helps tremendously on NA Birds, where a high capacity to learn is required to do well.

Method	Stan. Dogs	Ox. Pets	CUB 200	NA Birds	CIFAR 10	Caltech 101	Mean PC
LEEP	14.6 ± 0.2	37.3 ± 0.5	46.3 ± 0.1	24.0 ± 0.2	29.6 ± 0.4	9.3 ± 0.3	26.8 ± 0.1
NCE	18.4 ± 0.2	20.9 ± 0.2	48.5 ± 0.3	51.9 ± 1.4	16.4 ± 1.4	13.4 ± 0.3	28.3 ± 0.5
HScore	24.5 ± 27.9	22.7 ± 12.5	40.6 ± 27.7	26.7 ± 16.3	2.4 ± 9.5	36.4 ± 15.2	25.6 ± 9.2
1-NN CV	79.7 ± 0.6	70.6 ± 1.5	72.3 ± 1.6	75.5 ± 1.6	43.9 ± 1.9	66.0 ± 0.6	68.0 ± 0.7
5-NN CV	79.5 ± 0.5	70.6 ± 1.3	74.9 ± 2.1	78.0 ± 2.3	42.1 ± 0.7	65.8 ± 0.6	68.5 ± 0.7
1-NN	79.1 ± 1.0	70.1 ± 2.4	70.6 ± 2.2	73.6 ± 2.4	43.8 ± 3.8	65.6 ± 1.1	67.1 ± 0.8
5-NN	79.8 ± 1.5	67.8 ± 2.0	77.3 ± 1.0	77.4 ± 1.4	40.0 ± 4.4	65.7 ± 1.6	68.0 ± 1.2
Logistic	80.2 ± 0.7	74.6 ± 1.8	74.0 ± 2.4	74.7 ± 1.2	41.3 ± 2.9	70.4 ± 1.3	69.2 ± 1.1
RSA Resnet-50	70.8 ± 0.5	75.6 ± 0.4	67.7 ± 0.9	67.0 ± 0.8	44.2 ± 1.5	68.6 ± 0.3	65.7 ± 0.2
RSA Resnet-18	72.3 ± 0.7	76.3 ± 0.5	69.6 ± 1.0	64.5 ± 0.7	44.4 ± 1.5	68.3 ± 0.5	65.9 ± 0.4
RSA GoogLeNet	58.2 ± 0.5	65.8 ± 0.4	33.6 ± 1.8	45.0 ± 2.0	30.0 ± 1.6	69.0 ± 1.1	50.3 ± 0.4
RSA Alexnet	-15.0 ± 1.7	41.3 ± 1.5	6.0 ± 1.6	-17.1 ± 1.2	46.8 ± 0.5	28.4 ± 1.1	15.1 ± 0.6
RSA Full	26.1 ± 0.7	73.3 ± 0.5	-7.4 ± 0.8	-42.0 ± 1.1	28.6 ± 1.6	50.8 ± 0.8	21.6 ± 0.4
DDS Resnet-50	70.0 ± 0.5	75.4 ± 0.4	67.0 ± 0.7	63.9 ± 0.4	44.4 ± 0.9	68.9 ± 0.3	64.9 ± 0.2
DDS Resnet-18	71.4 ± 0.6	76.1 ± 0.4	67.6 ± 0.8	62.0 ± 0.4	43.9 ± 1.2	69.0 ± 0.3	65.0 ± 0.3
DDS GoogLeNet	59.5 ± 0.7	68.5 ± 0.4	39.4 ± 1.2	46.3 ± 1.2	36.2 ± 0.8	69.1 ± 0.8	53.2 ± 0.4
DDS Alexnet	-11.0 ± 1.1	45.4 ± 1.4	10.7 ± 1.4	-13.9 ± 1.2	47.2 ± 1.0	32.0 ± 0.7	18.4 ± 0.4
DDS Full	28.0 ± 0.5	73.6 ± 0.6	-1.6 ± 0.5	-35.2 ± 1.5	34.7 ± 1.5	50.8 ± 0.7	25.0 ± 0.4
PARC	73.8 ± 0.6	57.5 ± 2.0	75.9 ± 0.6	79.2 ± 0.8	42.9 ± 1.0	74.4 ± 0.7	67.3 ± 0.5

Table 9: **All Targets: Capacity to Change.** Pearson Correlation for each target dataset with the ℓ_s heuristic incorporated for a budget size of $n = 500$.



(a) Varying f on its own.

(b) Varying f with ℓ_s applied.

Figure 3: **Dimensionality Reduction.** In this figure, we vary the value of f used for dimensionality reduction with and without the ℓ_s heuristic applied. The left plot is the same as the one in the main paper except with PARC added. Note that the best choice for f is different with and without ℓ_s. We failed to consider that in the original submission and will fix that for the final version.

A.4 All Tweaks

In Tab. 10, we show the Pearson Correlation for each dataset individually for all feature-based methods with both feature reduction and the ℓ_s heuristic applied. These are all tweaks outlined in our paper, and they improve the performance on all datasets significantly over each method's original versions. We plot the means in Fig. 3.

Note that the best dimensionality for feature reduction changes for most methods when the layer heuristic is applied. Thus, in our paper we report different values of f in the final table than in the ablations for dimensionality reduction itself.

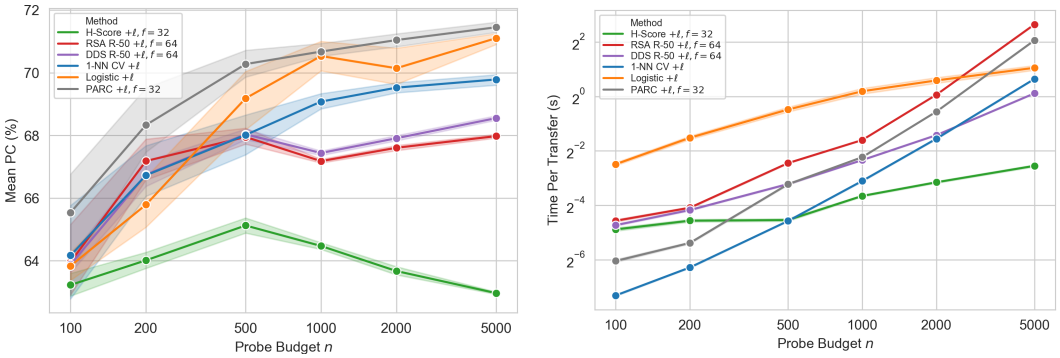
Method	Stan. Dogs	Ox. Pets	CUB 200	NA Birds	CIFAR 10	Caltech 101	Mean PC
HScore $f = 16$	77.0 ± 0.6	68.3 ± 0.5	74.6 ± 0.9	69.7 ± 0.8	31.4 ± 0.9	72.7 ± 0.5	65.6 ± 0.2
HScore $f = 32$	78.1 ± 0.5	66.3 ± 0.6	76.8 ± 0.5	72.1 ± 0.5	24.7 ± 0.4	72.7 ± 0.2	65.1 ± 0.3
HScore $f = 64$	76.6 ± 0.4	62.6 ± 0.7	77.6 ± 0.4	73.0 ± 0.5	18.7 ± 0.1	72.5 ± 0.4	63.5 ± 0.2
HScore $f = 128$	74.2 ± 0.5	54.1 ± 0.6	77.8 ± 0.2	74.4 ± 0.7	14.1 ± 0.1	70.8 ± 0.3	60.9 ± 0.2
HScore $f = 256$	70.0 ± 0.3	43.5 ± 0.3	76.7 ± 0.3	76.4 ± 0.4	11.4 ± 0.0	63.1 ± 0.0	56.9 ± 0.1
RSA R-50 $f = 16$	74.9 ± 1.2	72.3 ± 0.6	72.5 ± 0.8	67.3 ± 2.2	44.6 ± 0.8	66.8 ± 0.7	66.4 ± 0.6
RSA R-50 $f = 32$	77.2 ± 0.8	69.4 ± 0.4	78.0 ± 0.7	70.9 ± 0.5	48.8 ± 0.7	68.3 ± 0.4	68.8 ± 0.0
RSA R-50 $f = 64$	74.6 ± 0.4	64.3 ± 0.5	79.7 ± 0.6	76.9 ± 1.4	46.7 ± 0.5	65.5 ± 0.4	67.9 ± 0.3
RSA R-50 $f = 128$	70.3 ± 0.4	58.9 ± 0.3	77.9 ± 1.2	77.5 ± 0.8	43.0 ± 0.4	61.5 ± 0.2	64.9 ± 0.1
RSA R-50 $f = 256$	63.2 ± 0.5	53.6 ± 0.6	72.6 ± 1.1	67.0 ± 1.4	36.2 ± 0.7	59.2 ± 0.5	58.6 ± 0.6
DDS R-50 $f = 16$	74.1 ± 0.8	71.7 ± 0.4	71.7 ± 0.5	66.2 ± 1.0	44.6 ± 0.5	68.2 ± 0.8	66.1 ± 0.3
DDS R-50 $f = 32$	76.0 ± 0.5	71.2 ± 0.5	76.2 ± 0.6	68.4 ± 0.4	47.7 ± 1.3	70.0 ± 0.3	68.2 ± 0.2
DDS R-50 $f = 64$	73.9 ± 0.3	69.0 ± 0.4	77.4 ± 0.6	71.0 ± 0.9	47.1 ± 0.7	69.9 ± 0.3	68.0 ± 0.2
DDS R-50 $f = 128$	70.1 ± 0.2	64.1 ± 0.2	76.7 ± 1.0	69.0 ± 0.2	44.6 ± 0.5	68.9 ± 0.3	65.6 ± 0.1
DDS R-50 $f = 256$	64.1 ± 0.4	57.4 ± 0.5	73.2 ± 0.6	61.5 ± 1.0	38.4 ± 0.3	65.6 ± 0.3	60.0 ± 0.4
1-NN CV $f = 16$	78.5 ± 1.2	70.9 ± 1.5	71.3 ± 1.4	72.0 ± 2.4	44.3 ± 2.5	65.4 ± 0.9	67.1 ± 1.0
1-NN CV $f = 32$	78.7 ± 0.4	71.2 ± 1.2	71.4 ± 1.3	74.4 ± 2.3	44.5 ± 2.0	66.4 ± 1.1	67.8 ± 0.8
1-NN CV $f = 64$	79.1 ± 0.7	71.7 ± 1.6	72.2 ± 1.5	74.6 ± 2.1	43.7 ± 2.1	67.0 ± 0.7	68.0 ± 0.7
1-NN CV $f = 128$	79.3 ± 0.7	71.4 ± 1.4	72.2 ± 1.4	74.9 ± 1.7	43.2 ± 1.2	67.1 ± 0.5	68.0 ± 0.7
1-NN CV $f = 256$	79.7 ± 0.6	70.6 ± 1.3	72.2 ± 1.7	75.3 ± 1.7	44.0 ± 1.5	66.6 ± 0.5	68.1 ± 0.7
Logistic $f = 16$	78.8 ± 1.2	71.5 ± 1.3	70.6 ± 1.4	70.9 ± 2.5	39.8 ± 4.3	67.1 ± 0.7	66.5 ± 1.0
Logistic $f = 32$	80.3 ± 0.5	72.3 ± 2.1	72.4 ± 1.6	73.0 ± 2.9	37.6 ± 3.8	68.7 ± 0.3	67.4 ± 1.3
Logistic $f = 64$	80.4 ± 1.0	73.1 ± 1.5	73.1 ± 1.4	74.8 ± 2.6	40.1 ± 3.1	70.0 ± 2.2	68.6 ± 1.2
Logistic $f = 128$	80.6 ± 1.0	73.1 ± 2.1	73.3 ± 1.6	74.4 ± 1.9	41.2 ± 2.0	70.7 ± 1.0	68.9 ± 1.1
Logistic $f = 256$	80.8 ± 0.8	73.4 ± 2.1	73.8 ± 1.8	74.6 ± 1.9	40.8 ± 2.5	70.6 ± 1.1	69.0 ± 1.3
PARC $f = 16$	76.9 ± 1.2	61.1 ± 3.6	75.8 ± 0.9	76.4 ± 0.7	45.1 ± 1.9	76.6 ± 1.0	68.6 ± 0.5
PARC $f = 32$	78.3 ± 0.4	66.9 ± 2.2	76.3 ± 1.4	77.2 ± 0.8	45.1 ± 1.6	78.0 ± 0.5	70.3 ± 0.5
PARC $f = 64$	78.4 ± 0.7	67.2 ± 2.2	76.1 ± 1.3	76.5 ± 1.8	43.6 ± 1.2	78.7 ± 1.0	70.1 ± 0.8
PARC $f = 128$	78.1 ± 1.2	64.8 ± 2.3	76.6 ± 0.7	74.6 ± 2.0	38.5 ± 1.6	77.9 ± 0.7	68.4 ± 0.4
PARC $f = 256$	79.0 ± 0.9	59.6 ± 2.8	76.1 ± 1.5	72.5 ± 2.4	30.3 ± 1.4	75.4 ± 1.1	65.5 ± 0.7

Table 10: **All Targets: All Tweaks.** Pearson Correlation for each target dataset for each feature-based method with PCA reduction down to f features and with the ℓ_s heuristic incorporated for a budget size of $n = 500$.

B More Results for Varying Probe Set Size

In the main paper, we only varied the probe set size for the original version of each method. Because the original methods all had vastly different correlations and time taken, we had to plot relative correlation and the ratio of time taken instead. This showed how different methods scaled differently with extra data, but it didn't show how well each method performed relative to each other. In Fig. 4, we show the performance and time taken while varying the probe set size n for each competitive method with all beneficial tweaks applied. These plots are absolute so that the performance of each method can be compared.

PARC outperforms all other methods for all values of n in this setting, though using more data requires exponentially more time. This is mostly because PARC computes correlations on an $n \times n$ matrix, meaning it scales poorly with n. Logistic scales better in time taken than PARC, becoming faster at $n = 5000$. While it's still slightly worse than PARC, it's a good alternative to use if you plan to use the full dataset, for instance.



(a) Mean PC of each method while varying n.

(b) Time taken by each method while varying n.

Figure 4: **Varying Probe Set Size.** In this figure, we vary the probe set size n for the best version of each competitive method (with all useful tweaks applied). Note that the absolute performance is plotted here, not relative like in the main paper.

C Other Ways to Model Capacity to Change

In Section 4 of the main paper, we ensemble a heuristic with each method according to the following equation:

$$(\alpha_s^t)'_1 = \frac{\alpha_s^t - \mu^t}{\sigma^t} + \frac{\ell_s}{\ell_{\text{max}}} \tag{9}$$

This takes the form of normalizing the method's output scores and then adding the ratio between the number of layers in the network ℓ_s and the number of layers in the deepest network ℓ_{max}. For all experiments, we used $\ell_{\text{max}} = 50$, because the deepest network we used was ResNet-50.

In this section, we test more ways to perform this heuristic ensemble. Since all approaches output a different range of scores, it was important to first normalize the output scores to be consistent across methods. In the original paper, we subtract by the mean and divide by the standard deviation over all source models to accomplish this normalization. However, another way to normalize the scores would be to re-scale the empirical min to 0 and empirical max to 1.

$$(\alpha_s^t)'_2 = \frac{\alpha_s^t - \min^t}{\max^t - \min^t} + \frac{\ell_s}{\ell_{\text{max}}} \tag{10}$$

This is a more intuitive form of normalization, but it doesn't perform as well as Eq. 9 because it puts too much weight on the number of layers. To address this, we can simply scale the heuristic term:

$$(\alpha_s^t)'_3 = \frac{\alpha_s^t - \min^t}{\max^t - \min^t} + \lambda_\ell \frac{\ell_s}{\ell_{\text{max}}} \tag{11}$$

We find $\lambda_\ell = \frac{1}{4}$ to be a good choice. In Tab. 11, we benchmark the performance of these ensembling equations for each method. Eq. 9 works the best for all methods (within error bars), with Eq. 11 working similarly well for around half the methods. Note that Eq. 10 works the best only for LEEP and NCE, indicating that their distribution of outputs is very different to that of the other methods (which makes sense given that they both output a log-based score). We could find a value for λ_ℓ that maximizes the performance of each method, but this risks overfitting to our benchmark. We leave hyperparemeter tuning on a separate benchmark to future work. Thus, because it requires no hyperparameter tuning and works on a broad range of methods, we use Eq. 9 in our original paper.

Method	Mean PC (%) Original	Mean PC (%) Eq. 9	Mean PC (%) Eq. 10	Mean PC (%) Eq. 11
LEEP	10.82 ± 0.13	26.83 ± 0.12	$\mathbf{47.05 \pm 0.09}$	26.03 ± 0.12
NCE	2.08 ± 0.67	28.27 ± 0.47	$\mathbf{48.15 \pm 0.14}$	29.75 ± 0.45
H-Score $f = 32$	55.95 ± 0.55	$\mathbf{65.12 \pm 0.28}$	60.44 ± 0.12	$\mathbf{65.08 \pm 0.29}$
RSA R-50 $f = 64$	66.17 ± 0.47	$\mathbf{67.95 \pm 0.28}$	61.24 ± 0.09	67.42 ± 0.23
DDS R-50 $f = 64$	63.90 ± 0.37	$\mathbf{68.04 \pm 0.20}$	62.80 ± 0.05	$\mathbf{68.08 \pm 0.17}$
1-NN CV	60.75 ± 1.25	$\mathbf{68.01 \pm 0.75}$	63.95 ± 0.41	$\mathbf{67.95 \pm 0.76}$
Logistic	61.94 ± 1.42	$\mathbf{69.18 \pm 1.10}$	64.60 ± 0.53	$\mathbf{69.11 \pm 1.13}$
PARC $f = 32$	59.31 ± 0.68	$\mathbf{70.28 \pm 0.47}$	60.80 ± 0.19	69.75 ± 0.42

Table 11: **Adding Capacity to Change.** Different ways to incorporate the number of layers heuristic ℓ_s/ℓ_{max}. The best ensembling methods (and those within one standard deviation of the best) for each method are shown in bold for each method. For all competetive methods, Eq. 9 works the best, so that is what we use in the original paper.

D Sources for the Crowd-Sourced Benchmark

In Tab. 12 we list all source models use in our crowd-sourced benchmark (Sec. 6.2 in the original paper). To extract features from each ResNet-based architecture, we simply globally pool the features in the C5 layer. Also, while the crowd-sourced models contain ResNet-101 with $\ell_s = 101$, we still fix $\ell_{\text{max}} = 50$ for Eq. 9 to be consistent with the original benchmark.

Method	Backbone	Dataset	Model ID	License	Source
Faster R-CNN	ResNet-101 C4	COCO	138204752	CC BY-SA 3.0	Detectron v2
Faster R-CNN	ResNet-50 C4	COCO	137257644	CC BY-SA 3.0	Detectron v2
Faster R-CNN	ResNet-50 C4	COCO	137849393	CC BY-SA 3.0	Detectron v2
Faster R-CNN	ResNet-50 C4	VOC 07+12	142202221	CC BY-SA 3.0	Detectron v2
Faster R-CNN	ResNet-50 FPN	COCO	137257794	CC BY-SA 3.0	Detectron v2
Faster R-CNN	ResNet-50 FPN	COCO	137849458	CC BY-SA 3.0	Detectron v2
Faster R-CNN	ResNet-101 FPN	COCO	137851257	CC BY-SA 3.0	Detectron v2
Mask R-CNN	ResNet-101 FPN	COCO	138205316	CC BY-SA 3.0	Detectron v2
Mask R-CNN	ResNet-101 C4	COCO	138363239	CC BY-SA 3.0	Detectron v2
Mask R-CNN	ResNet-50 C4	COCO	137259246	CC BY-SA 3.0	Detectron v2
Mask R-CNN	ResNet-50 C4	COCO	137849525	CC BY-SA 3.0	Detectron v2
Mask R-CNN	ResNet-50 FPN	COCO	137260431	CC BY-SA 3.0	Detectron v2
Mask R-CNN	ResNet-50 FPN	COCO	137849600	CC BY-SA 3.0	Detectron v2
Mask R-CNN	ResNet-50 FPN	Cityscapes	142423278	CC BY-SA 3.0	Detectron v2
Mask R-CNN	ResNet-50 FPN	LVIS	144219072	CC BY-SA 3.0	Detectron v2
Mask R-CNN	ResNet-101 FPN	LVIS	144219035	CC BY-SA 3.0	Detectron v2
Keypoint R-CNN	ResNet-101 FPN	COCO	138363331	CC BY-SA 3.0	Detectron v2
Keypoint R-CNN	ResNet-50 FPN	COCO	137261548	CC BY-SA 3.0	Detectron v2
Keypoint R-CNN	ResNet-50 FPN	COCO	137849621	CC BY-SA 3.0	Detectron v2
Panoptic R-CNN	ResNet-101 FPN	COCO	139514519	CC BY-SA 3.0	Detectron v2
Panoptic R-CNN	ResNet-101 FPN	COCO	139797668	CC BY-SA 3.0	Detectron v2
Panoptic R-CNN	ResNet-50 FPN	COCO	139514544	CC BY-SA 3.0	Detectron v2
Panoptic R-CNN	ResNet-50 FPN	COCO	139514569	CC BY-SA 3.0	Detectron v2
RetinaNet	ResNet-101	COCO	190397697	CC BY-SA 3.0	Detectron v2
RetinaNet	ResNet-50	COCO	190397773	CC BY-SA 3.0	Detectron v2
RetinaNet	ResNet-50	COCO	190397829	CC BY-SA 3.0	Detectron v2
SimCLR	ResNet-101	ImageNet 1k		MIT	VISSL
ClusterFit	ResNet-50	ImageNet 1k		MIT	VISSL
DeepCluster v2	ResNet-50	ImageNet 1k		MIT	VISSL
Jigsaw	ResNet-50	ImageNet 22k		MIT	VISSL
MOCO	ResNet-50	ImageNet 1k		MIT	VISSL
NPID	ResNet-50	ImageNet 1k		MIT	VISSL
PIRL	ResNet-50	ImageNet 1k		MIT	VISSL
RotNet	ResNet-50	ImageNet 22k		MIT	VISSL
SimCLR	ResNet-50	ImageNet 1k		MIT	VISSL
SWAV	ResNet-50	ImageNet 1k		MIT	VISSL
Semi-Supervised	ResNet-50	Instagram		MIT	VISSL
Semi-Supervised	ResNet-50	YFCC100M		MIT	VISSL
Supervised	ResNet-50	Places205		MIT	VISSL

Table 12: **Crowd-Sourced Source Models.** For our crowd-sourced benchmark, we download pretrained source models from two model banks. For each source model, we list the originating method, the architecture it uses, the dataset it was trained on, the corresponding model ID, the license the source model was released under, and the source model bank. For models without an ID (i.e., VISSL models), we take the highest performing model released.

E A Note on Metrics

Is Pearson Correlation a suitable metric. We believe that Pearson Correlation (instead of some top-k accuracy) is the right metric to use for Scalable Diverse Model Selection for several reasons:

1. We want our results to be meaningful no matter the model bank being used. If we just looked at the top model output, our results would be entirely invalidated if models were added or removed to the model bank. We care about the intrinsic quality of the selection models themselves, rather than specifically how they perform on our exact model bank.
2. Our goal is to address diverse model selection, meaning that the source models should vary in architecture, dataset, and task. If we only look at the top model returned by the selection algorithm, we're not evaluating how well that method can compare across these source model variations. In order to do that, we need to take the ranking of all the models into account.
3. Practitioners using these model selection algorithms will have different needs depending on their situation. If they need to run their models on phones or other low-power devices, ResNet-50 might not be an option. They'd want to compare e.g., ResNet-18, GoogLeNet, and MobileNet among other more optimized architectures, meaning our evaluation method must take that into account.

Other Metrics. Nevertheless, it would be useful to have additional metrics that depend more on the actual top results predicted by the method. For this purpose, we think relative accuracy would be the easiest to interpret. That is, take the top-k models suggested by the algorithm, fetch their transfer performance and average those numbers. Finally, divide this average by the transfer performance of the best model to obtain a notion of "model selection accuracy". Here are the results for different values of k. We list columns with the original performance of each method (w/o the tricks discussed in the paper), as well as the same with the tricks added (w/ Tricks).

Method	Original, $k = 1$	Original, $k = 3$	w/ Tricks, $k = 1$	w/ Tricks, $k = 3$	w/ Tricks, $k = 5$
LEEP	$93.18\% \pm 0.00$	$90.72\% \pm 0.04$	$99.56\% \pm 0.00$	$92.79\% \pm 0.02$	$90.86\% \pm 0.00$
NCE	$95.80\% \pm 1.65$	$89.66\% \pm 1.62$	$98.82\% \pm 0.40$	$96.80\% \pm 0.26$	$94.56\% \pm 0.25$
HScore	$84.78\% \pm 2.77$	$83.55\% \pm 2.17$	$99.46\% \pm 0.10$	$98.63\% \pm 0.04$	$97.66\% \pm 0.14$
RSA R-50	$98.27\% \pm 0.12$	$98.32\% \pm 0.15$	$99.35\% \pm 0.04$	$98.66\% \pm 0.03$	$97.75\% \pm 0.11$
DDS R-50	$99.37\% \pm 0.00$	$98.25\% \pm 0.17$	$99.37\% \pm 0.01$	$98.62\% \pm 0.03$	$97.70\% \pm 0.09$
1-NN CV	$99.60\% \pm 0.05$	$97.07\% \pm 0.79$	$99.60\% \pm 0.05$	$97.75\% \pm 0.27$	$96.94\% \pm 0.02$
Logistic	$99.58\% \pm 0.03$	$96.46\% \pm 0.92$	$99.45\% \pm 0.17$	$97.86\% \pm 0.26$	$97.08\% \pm 0.27$
PARC	·	·	$99.31\% \pm 0.18$	$98.43\% \pm 0.12$	$97.88\% \pm 0.00$

Table 13: **Top-K Relative Accuracy.** The accuracy of the predicted model divided by the accuracy of the best possible model averaged across the top k models for each modle selection method. This only considers the top models, but this gives an estimate of how well each method does in terms of comparing raw accuracy.

Three points are clear from these results:

1. Adding the tricks described in the paper significantly improves the top-k results for almost all methods. This seems to be true no matter what mode of analysis we use.
2. With the tricks applied, some methods like LEEP have very high top-1 accuracies, but quickly fall off as more than 1 model is taken into account. Top-k results like these are inherently flawed in that they depend significantly on the list of models used, as small changes in that list can quickly shake up which algorithm performs the best, which is alleviated by a metric like Pearson Correlation which takes all models into account.
3. While PARC w/ Tricks isn't explicitly at the top except for $k \geq 5$, it's still a very strong contender that's fast and can be applied to any source task, architecture, or target task. Moreover, it's well calibrated for all types of models (which is what the Pearson Correlation results show).

We'd like to reiterate that this metric only tests 1-5 of the top selected models and does not test whether the algorithm is robust across source, architecture, or task (as those variations aren't likely to come up in the top 5 models). Thus, we keep Pearson Correlation as our main metric for evaluation, since top-k metrics don't capture our goal of testing a method's robustness to model diversity.