Guided Recommendation for Model Fine-Tuning

Hao Li, Charless Fowlkes, Hao Yang, Onkar Dabeer, Zhuowen Tu, Stefano Soatto
AWS AI Labs
{haolimax, fowlkec, haoyng, onkardab, ztu, soattos}@amazon.com

Abstract

Model selection is essential for reducing the search cost of the best pre-trained model over a large-scale model zoo for a downstream task. After analyzing recent hand-designed model selection criteria with 400+ ImageNet pre-trained models and 40 downstream tasks, we find that they can fail due to invalid assumptions and intrinsic limitations. The prior knowledge on model capacity and dataset also can not be easily integrated into the existing criteria. To address these issues, we propose to convert model selection as a recommendation problem and to learn from the past training history. Specifically, we characterize the meta information of datasets and models as features, and use their transfer learning performance as the guided score. With thousands of historical training jobs, a recommendation system can be learned to predict the model selection score given the features of the dataset and the model as input. Our approach enables integrating existing model selection scores as additional features and scales with more historical data. We evaluate the prediction accuracy with 22 pre-trained models over 40 downstream tasks. With extensive evaluations, we show that the learned approach can outperform prior hand-designed model selection methods significantly when relevant training history is available.

1. Introduction

Much of the success of deep learning can be ascribed to its flexibility: One can train a neural network on a task, and then use it on a different one, typically after fine-tuning. There are currently two trends for scaling this practice: One is to pre-train a large number of specialized models (a "Model Zoo" [10]) and then select one to fine-tune once the downstream task of interest becomes manifest, typically with a smaller fine-tuning dataset. Another is to pre-train a single "Foundation Model" which is then used to support any and all downstream tasks [47, 57].

Without additional specifications, the second case is a subset of the first, for one can take the Model Zoo and Model Selection (MS) mechanism and call it a single model. For this reason, Foundation Models are characterized as *homogeneous and task-agnostic*, where homogeneity refers to a single neural network architecture, in contrast with the *heterogeneous* collection of models in a zoo. Even with this restriction, the model zoo is more general, for nothing prevents a Foundation Model to be part of a zoo. In addition, selecting a smaller dedicated model pretrained for a task can be much more efficient than using a giant monolithic model For these reasons, we focus on model selection over a large heterogeneous model zoo for fine-tuning as the key solution for scaling inference to a wide variety of downstream tasks.

Brute-force model selection [1, 12] requires fine-tuning each pre-trained model on the task of interest, and then ranking them using the test error on a held-out dataset as a *model selection score*. This is not feasible for large model zoos. Current model-selection methods therefore aim to *predict* the model selection score without actually fine-tuning.

However, current model selection methods do not take into explicit account even basic characteristics of the fine-tuning dataset, such as the number of classes or the number of images, nor of the pre-trained model, such as the model family, the size of the input, the number of parameters and the dataset on which it is pre-trained. While coarse, these features can affect the best model to fine-tune, since a mismatch between fine-tuning dataset size and pre-trained model, or input dimensions, or number of classes, can influence the success of downstream performance.

Instead of proposing yet another model selection score, *we propose re-framing model selection as a recommender system*, and directly predict the selection score and corresponding ranking, from whatever existing model selection scores are readily available, in addition to whatever coarse features a user deems informative – which may be context dependent, as some users may wish to penalize large models, or models that require high-resolution input. Such features help *guide* the model selection using criteria beyond raw downstream validation error. For this reason, we refer to our recommendation approach as *guided*, in addition to *trained*.

We find that incorporating model size, dataset size, cardinality of the hypothesis set and other simple features already improves the prediction of the expected model selection score compared to current model selection methods. Coarse features, such as the model class family (convolutional, fully-connected, residual, attention-based, etc.) can help associate certain architectural inductive biases such as translation vs. permutation invariance, to the best-fitting downstream tasks, for instance object detection vs. image inpainting or segmentation.

Our contribution can be summarized as:

- We conduct comprehensive analysis of existing model selection approaches with a large heterogeneous model zoo and confirmed their limitations. We find feature-based model selection becomes inaccurate when the target dataset is different from the source task and the effect of model initialization diminishes as the number of images grows. The useful meta information and prior knowledge in the training history are often neglected and cannot be easily integrated into existing model selection criteria.
- We convert the model selection problem as model recommendation by learning from past training history. The meta information of both dataset and model are embedded as features and a recommender system can be learned to predict the performance. The existing model selection can be used as additional features and makes the framework comply with existing approaches. We show significant performance improvement over traditional model selection methods when historical training data is available and relevant.

In the next section we formalize the MS problem, and discuss the issues with existing approaches. In section 3 we describe our approach to casting it as a recommendation system and evaluate it in the following section.

2. Background

2.1. Problem Formalization

Let T_i be pre-training candidate tasks, with $i = 1, \dots, M$, encoded in their corresponding datasets $D_i = \{(x_k, y_k)\}_{k=1}^{N_i}$ (the dataset is all a model knows about the task prior to training), used to train a chosen architecture (function class) $\phi_i(\cdot; w_i)$ by minimizing a loss function L_i with respect to the weights w, yielding

$$w_i = \arg\min_w \underbrace{\sum_{(x_k,y_k)\in D_i} \ell(y_k, \phi_i(x_k; w))}_{L_i} \doteq \hat{w}(\phi_i, D_i) \tag{1}$$

where the pre-trained weights w_i are a function $\hat{w}$ of the dataset, the architecture, and the pointwise loss ℓ, which is typically cross-entropy, in addition to the optimization procedure, regularizers, hyperparameters, and other factors that we omit for simplicity since we wish to focus on the relative role of the architecture and the dataset.

When fine-tuning a model ϕ_i for a different task $T_{j\neq i}$, the architecture is conditioned on using ϕ_i, either as a frozen embedding, or as an initialization, so the model to be fine-tuned for the task T_j using the dataset D_j, has the form $\phi_j(\phi_i(\cdot; w_i); w)$ and the fine-tuning loss $\tilde{V}_{ij}$ is

$$\min_w \sum_{(x_k,y_k)\in D_j} \ell(y_k, \phi_j(\phi_i(x_k; w_i); w)) \doteq \tilde{V}(\phi_i|\phi_j, D_i, D_j) \tag{2}$$

$\tilde{V}_{ij}$ is the *empirical model selection score*, corresponding to the training error during fine-tuning. Brute-force model selection consists of solving

$$\hat{\phi}_i = \arg\min_{\phi_i} \tilde{V}(\phi_i \mid \phi_j, D_i, D_j). \tag{3}$$

It is immediate to see that for the function $\hat{\phi}_i$ to be constant with respect to ϕ_j (that is, for the pre-trained representation to be *task-agnostic*) it would either have to be conditioned on all possible tasks (including those with different hypothesis spaces, hence be *non-homogeneous*), or be a *trivial* lossless compression of the data, for the task could turn out to be reproducing an identical copy of the data. This would defer the burden of learning to the fine-tuning phase, annihilating the value of pre-training and undermining the main premise of Foundation Models as homogeneous *and* task-agnostic *and* optimal for fine-tuning. This further reinforces our focus on heterogeneous model selection.

The validation error on a held-out dataset, or ideally the marginal over all possible fine-tuning datasets, is the (expected) *model selection score*

$$\hat{V}_{ij} = \mathbb{E}_{D_j} \tilde{V}(\phi_i|\phi_j, D_i, D_j) \tag{4}$$

which is clearly not computable.

Feature-based Model Selection Most recent MS methods (e.g., LFC [10], PARC [5], and LogME [60]) extract features with each candidate model on the target dataset, and then calculate the MS score. Given a dataset $D = \{\mathbf{x}, \mathbf{y}\}$, let $f_w(x_i)$ denote the feature vector extracted from penultimate layer of pre-trained model ϕ for data x_i. The LFC score is calculated as

$$S_{\mathbf{LFC}}(\mathbf{x}, \mathbf{y}) = f_w(\mathbf{x}) f_w(\mathbf{x})^T \cdot \mathbf{y}\mathbf{y}^T \tag{5}$$

where $(\mathbf{y}\mathbf{y}^T)_{i,j} = 1$ if x_i and x_j have the same label and -1 otherwise. The normalized S_{LFC} can be interpreted as the Pearson Correlation between the features and labels. The assumption is that a better candidate model's initialization usually has similar features for samples with the same labels.

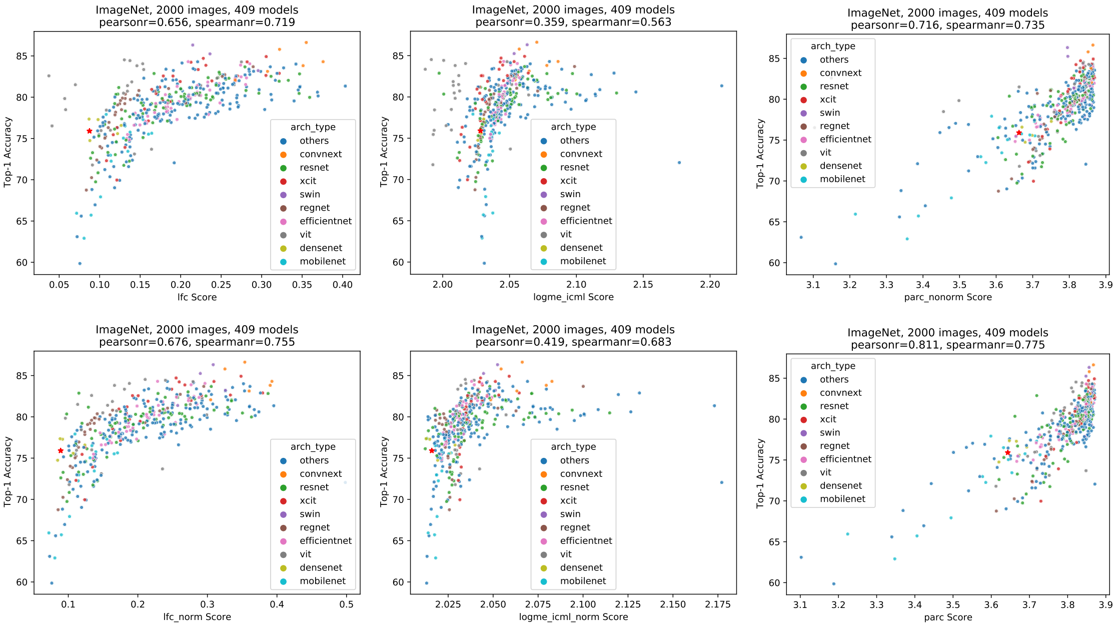


Figure 1. Comparison of MS algorithms with 400+ ImageNet pre-trained models. In the 1st row, features are not normalized. LFC [10], LogME [60] and PARC [5] (w.o. normalization) all treat ViTs (gray points) as outliers. When features are normalized in the 2nd row, MS scores improve for all methods and ViTs are not outliers, indicating the importance of feature normalization for heterogeneous models.

2.2. Model Selection Limitations

Feature-based MS methods usually fix the candidate model as feature extractors and assume the fine-tuning process does not change the backbone weights too much. However, the assumption may not hold in practice and results in failure. Here we evaluate three MS algorithms (LFC [10], PARC [5], and LogME [60]) with two settings and demonstrate the cases in which they can fail. a) fine-tuning the 400+ ImageNet pre-trained models on ImageNet. The fine-tuning performance should be consistent with their pre-training performance. b) we select 22 models near the Pareto frontier of the 400+ models and fine-tune them on 40 downstream datasets. More setup details can be referred to Sec. 4.1.

Difficulty with heterogeneous models Existing MS methods usually validate their approaches with a homogeneous model zoo in which models differ only in pre-trained domains. And it is often believed that better ImageNet model also transfer better on downstream tasks [31], which seems to makes the problem of MS with heterogeneous model zoo trivial. However, we find that the optimal architecture or Pareto front models can be task dependent, which relies on both the inductive bias of the model and the characteristics of the dataset. In addition, existing MS algorithms can fail to accommodate new architectures such as ViTs which have much smaller feature dimensions compared to ResNets. As shown in Fig. 1, ViTs are outliers for MS methods without explicit normalization. Normalizing the input features of all architectures can mitigate this issue and improve the Pearson correlation scores. This was observed in [5] in which they further improve PARC by applying PCA and adding normalized network depth to incorporate the network capacity. However, the heuristic cannot always generalize across architectures, e.g., ViTs that come without the same depth concept in terms of convolutional layers. More details on the heterogeneous model zoo can be found in Appendix D.

Difficulty with dissimilar target datasets Existing MS algorithms often assume that the weights of pre-trained models do not change much during fine-tuning, which is valid for few-shot or linear probing where the majority of weights do not change much. However, the effect of initialization often diminishes as the dataset size grows. When training data is large, a random initialized model with high capacity can yield better performance than a pre-trained simple model. The MS score of the random initialized model can be lower than the pre-trained one, which does not represent the underlying generalization ability of the model. Fig. 2 shows a clear difference for the MS performance between *dogs* [30] and *aircrafts* [38]. Note that *dogs* is known to be similar to ImageNet but not *aircrafts*[1]. It verifies that MS can fail to predict top performing models when a downstream task is much different from the source task.

[1] Stanford Dogs (*dogs*) was built using images from ImageNet.

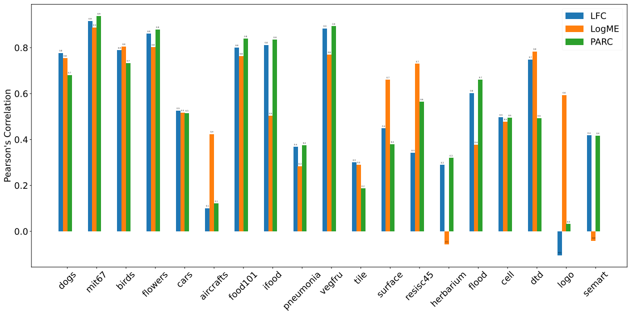


Figure 2. Pearson Correlation of three MS methods on the 19 fine-grained datasets with 22 models. Note that the correlation scores for *aircrafts* and *pneumonia* are much lower than other datasets.

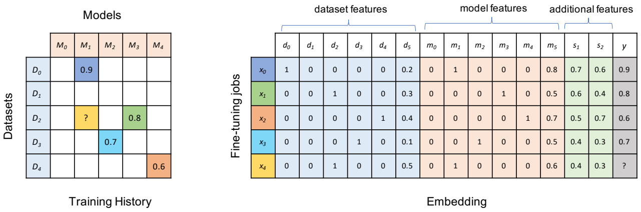
Difficulty of incorporating prior knowledge The inductive bias can be heuristically added to existing MS score, e.g. PARC [5] incorporates the model depth on top of the original MS score. However, adding such heuristics to existing MS score requires ad-hoc tuning of the scale of the heuristic score, which is hard to extend to more indicators. On the other hand, the importance of the model inductive biases is often associated with the dataset characteristics, which is hard to integrate manually, e.g., "a random initialized model with large capacity can generalize better than a small pre-trained model for a large dataset" and "a small model can perform the same as a bigger model for a simple task". In such cases, the effectiveness of model pre-training and capacity is also determined by the dataset size and task hardness. Therefore we need to associate the inductive bias of the model and the characteristics of the task. Existing MS methods often only use a small probe set with fixed number of images to reduce the computation cost, which neglects the meta information of target dataset size.

3. Learning to Recommend

Instead of manually designing a model selection criteria, we propose to learn to select models from the training history. Given the historical training results, we can characterize the features of datasets and models, and use fine-tuning performance as the ground truth. The goal is to predict performance on the target dataset for a given model. Then a model selector can be learned to select the optimal model for a given task. Fig. 3 illustrates the recommendation problem and how the training history is represented in the embedding space.

3.1. Model Selection as Recommendation

In order to frame model selection as a recommendation system, we represent the pre-trained model ϕ_i with an element of a vector space $\mathbf{v}_i$, and/or a simpler vectorized version of coarse features such as the number of parameters, input dimension, number of classes, index of the architecture family and pre-training dataset, etc. Similarly, we embed the



	M_0	M_1	M_2	M_3	M_4
D_0		0.9			
D_1					
D_2		?		0.8	
D_3			0.7		
D_4					0.6

	d_0	d_1	d_2	d_3	d_4	d_5	m_0	m_1	m_2	m_3	m_4	m_5	s_1	s_2	y
x_0	1	0	0	0	0	0.2	0	1	0	0	0	0.8	0.7	0.6	0.9
x_1	0	0	1	0	0	0.3	0	0	0	1	0	0.5	0.6	0.4	0.8
x_2	0	0	0	0	1	0.4	0	0	0	0	1	0.7	0.5	0.7	0.6
x_3	0	0	0	1	0	0.1	0	0	1	0	0	0.5	0.4	0.3	0.7
x_4	0	0	1	0	0	0.5	0	1	0	0	0	0.6	0.4	0.3	?

Figure 3. An illustration of learning from the training history and the representation of training data for the recommender system. The left matrix shows the training history with 4 pairs of dataset and model, with the goal to predict the performance of unknown pairs. The right figure shows the encoding of the each training job is concatenated features of the datset, model and others.

fine-tuning dataset D_j onto a set of features $\mathbf{v}_j$, for instance its cardinality and dimension of the hypothesis space. In addition, we can use any available predictive MS score U_{ij}. A recommendation system then implements a learnable map that, for each pre-trained model i and downstream task j predicts the expected score $\hat{V}_{ij}$:

$$\begin{aligned} \mathrm{RM} : [1, \dots, M] \times [1, \dots, J] &\rightarrow \mathbb{R} \\ (i, j) &\mapsto \hat{V}_{ij} = \mathrm{RM}(i, j) \end{aligned} \tag{6}$$

where $\hat{V}_{ij} = \psi_w(\mathbf{v_i}(\phi_i), \mathbf{v_j}(D_j), U_{ij})$ and ψ_w is a parameterized map, for instance a factorization machine, with learnable parameters w, trained to approximate the validation scores $\hat{V}_{ij}$. By choosing the features $\mathbf{v}_i, \mathbf{v}_j$ a user can factor in additional MS criteria besides the structure of the data, which is often captured in the MS scores U_{ij}, to guide the recommendation. Since the analytical expression for the functions $\hat{w}(\cdot)$, $\hat{\phi}(\cdot)$ and $\hat{V}(\cdot)$ are intractable, in this paper we study the problem in (3) empirically in the next section.

3.2. Recommendation Model

There are several options for learning the recommendation model. If the model zoo is fixed, a straight-forward solution is to learn a classifier that directly maps a given task to the best model [46]. The challenge lies at the usual insufficient training samples in comparison with the high dimensions of the dataset representation (e.g., 2048 for ResNet feature). And the fixed model zoo size makes it hard to adapt when new models are added to the model zoo. Therefore, we mainly consider the following two options:

- **Linear Regression (LR)** A LR model can be learned to predict performance with the concatenated task and model features. However, LR learns the effect of each feature independently and the interaction among features can not be modeled.
- **Factorization Machine (FM)**. Factorization Machine (FM) [49] is widely used in recommender systems and

CTR prediction. FM is often preferred over Linear Regression (LR) as it can learn the correlations among different features via latent embedding, even when there is no data point for the correlation. Given N historical transfer learning results $\{(\mathbf{d}_i, \mathbf{m}_i, \mathbf{y}_i)\}_{i=1}^{N}$, where $\mathbf{d}_i = \mathbf{v}(D) \in \mathbb{R}^d$ and $\mathbf{m}_i = \mathbf{v}(\phi) \in \mathbb{R}^m$ represents the feature embedding of dataset D and model ϕ for the ith fine-tuning job, d and m are their embedding length, y_i is the fine-tuning top-1 accuracy on the validation set. Let $\mathbf{z} \in \mathbb{R}^{d+m}$ denote the concatenated features of the pair of dataset and model, the predicted score of FM is

$$S_{\text{FM}}(\mathbf{z}) = w_0 + \sum_{i=1}^{|z|} w_i z_i + \sum_{i=1}^{|z|-1} \sum_{j=i+1}^{|z|} \langle \mathbf{u}_i, \mathbf{u}_j \rangle z_i z_j \quad (7)$$

where $\mathbf{u}_i \in \mathbb{R}^k$ is the latent representation of the ith feature. Note that the first two terms is actually LR. With the third term, FM considers interactions among features in addition to linear combination of features.

In the next sub-section, we will describe the feature embedding for datasets and models in detail.

3.3. Characterizing Datasets and Models

Dataset Embedding We explore the following descriptors for describing a task:

- **task difficulty**: If a task can be solved with a model's initial weights without much change, then the task is relatively easy for the model, e.g., linear probing (fixed embedding + SVM) is often used as a baseline for transfer learning. If a task gets high performance with linear probing, then it indicates the dataset is relatively easy to solve with a simple linear classifier. A MS score calculated with a fixed backbone (e.g., ResNet-18) can estimate the relative difficulty of the dataset.

- **number of samples**: The dataset size affects the task difficulty and model selection. A few-shot task is generally harder than tasks with large sizes and requires a strong model. The larger the dataset size, the more possibility of choosing a model without a strong initialization. Note that current MS algorithms (e.g., LFC [10]) usually use a prob set with fixed size, while in reality the prob set size could vary significantly.

- **number of classes**: When the total images are fixed, the task difficulty usually increase as the number of classes.

Model Embedding To characterize the model's inductive bias, we use the following features for model embedding:

- **architecture family**: architectures of the same family usually have similar inductive biases as they consist of similar modules, blocks and activation functions. We use the architecture family to categorize the inductive biases of models of the same family, such as ConvNeXt, ViT, Swin-Transformers, EfficientNet and etc.

- **input size**: it is reported higher resolution usually helps for downstream tasks [31], and we see this is true for fine-grained tasks (e.g., EfficientNet-B3 works best for *cars* and *aircrafts* as seen in Appendix D). On the other hand, simple cases (e.g., MNIST) may not benefit from higher resolutions.

- **model capacity**: a model with high capacity usually generalizes better with more data. This is measured by the number of parameters.

- **model complexity**: the calculation cost (GMACs) can represent the complexities.

- **pre-trained domain**: the pre-trained domain matters for the downstream task performance. If the source dataset is available, then the domain distance between the source domain and the target domain can be a indicator. However, such information is not always available. We have models pre-trained on ImageNet-1K and ImageNet-22K.

Additional Features The advantage of recommender system is that features related to the prediction can always be added, which makes the solution scalable to new features. Beyond the embedding of datasets and models, we can add additional features that are relevant to the performance prediction. The existing MS scores can be treated as additional features, as it considers the feasibility of the model's initial features (Eq. 5). Other features such as the semantic distance between the target task and the model's source task can also be added as additional feature, which could be useful for few-shot or zero-shot learning. We will leave this extension for future works.

4. Experiments

4.1. Settings

Datasets We collected three benchmarks and a total of 40 image classification tasks, including *19 fine-grained datasets*, *DomainNet* [45] and the *VTAB* [61]. Those datasets cover a wide range of domains and applications, such as *scenes*, *objects*, *food*, *texture*, *art* and *medical imaging*. DomainNet consists of 6 datasets of different domains with the same labels. VTAB consists of various tasks which can be categorized into *natural*, *structured* and *special*. More details about the datasets can be found in Appendix A.

Models The TIMM model zoo collected more than 550 ImageNet pre-trained models. We evaluated all pre-trained models and keep 409 models that can be fine-tuned with batch size 32 with a single v100 GPU. We evaluated their single image inference latency and identified the Pareto frontier in their latency-accuracy trade-off plot. We select 22 models that are near the Pareto Front curve, which covers a wide range of common architecture families, including ReseNet [17], DenseNet [21], MobileNet [20], EfficientNet [53], ViTs [11], Swin-T [36] and ConvNeXt [37]. The complete model list can be found in Appendix A.

Training History The models in the TIMM model zoo usually have ImageNet validation accuracy. Those results are obtained by training the model from scratch. If we reinitialize the last layer of each pre-trained model and fine-tune on ImageNet, we should expect the performance same with the reported results. In addition, we fine-tuned the selected 22 models end-to-end with HPO to obtain their best Top-1 accuracy on the 40 downstream tasks. All pre-trained models are trained with a single V100 GPU with the same range of hyperparameters. More details about the models and fine-tuning settings can be found in Appendix A.

Evaluation Metrics We measure MS performance on a given dataset (or probe set) with Pearson correlation coefficient, which measures linear correlation between MS score and oracle transfer performance, which is the covariance of the two variables divided by the product of their standard deviations. We use the mean Pearson correlation over all datasets in a benchmark to for comparison.

Recommendation Tasks We consider the following scenarios for model recommendation: a) Learn from the training history of one dataset with a subset of models and evaluate unseen models on the same dataset (e.g., ImageNet). b) Learn from the training history of one dataset and evaluate with the same models on unknown downstream tasks. c) Learning from the training history of both ImageNet and downstream tasks, and evaluate MS with known models on unseen tasks. Note that the diversity and amount of training samples (number of datasets and models) increases in these settings progressively. An illustration of the three settings can be seen in Fig. 4.

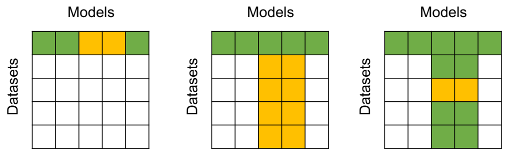


Figure 4. An illustration of the three evaluation settings. The green boxes represent the available training pairs of dataset and model with fine-tuning accuracy, and the yellow boxes indicate the pairs of dataset and model to be predicted for their performance.

4.2. Recommendation Results

4.2.1 Learning from the training history on ImageNet and evaluating unseen models on the same dataset

The ImageNet pre-trained model zoo provides off-the-shelf Top-1 accuracy for 400+ models, which can be used as the groundtruth performance. Note that given the volume of ImageNet, it is expected that models with or without ImageNet pre-training will converge to the same accuracy. To evaluate the learned MS on predicting the performance of unseen architectures on ImageNet, we randomly split the 400+ models with 80% of them for training and the rest 20% for evaluation. In Table 1, we compare the learned MS with different training features with the traditional feature-based MS methods.

With the pre-trained model weights, we see that learned MS (both LR and FM) with only meta features perform better than S_{LogME}. When the MS score is used as additional feature, the performance is better than using simply meta features or the MS score itself. When the models are randomly initialized, the feature-based MS methods fail to rank the models due to the randomness of extracted feature. In contrast, learning based method still get reasonable correlation score and is robust even when the noisy MS score is added as additional feature. The knowledge that larger models usually generalizes better can still be learned.

Note that since all training data are based on ImageNet training history, the dataset features are the same for all training data, and the correlation between dataset feature and model feature cannot be well learned. The learned MS score is mainly determined by the model feature, i.e., models with large capacity has better performance. Thus we see FM does not show advantage over a simple LR model, which is expected. We will see difference when expanding the training set in Sec. 4.2.3.

Table 1. MS learned with only ImageNet training history. The ImageNet benchmark samples 80% of the 409 models as training set while the rest of models are used for evaluation. The experiment is repeated 10 times and the mean/std values are reported.

Methods	Features	ImageNet	
		Pre-trained	Random Init.
feature-based	S_{LFC} [10]	0.65 ± 0.07	0.03 ± 0.10
	S_{LogME} [60]	0.35 ± 0.09	0.04 ± 0.08
	S_{PARC} [5]	**0.83** ± 0.04	0.08 ± 0.09
LR (ours)	$\mathbf{d}, \mathbf{m}$	0.53 ± 0.07	**0.57** ± 0.10
	$\mathbf{d}, \mathbf{m}, S_{\text{LFC}}$	0.73 ± 0.06	0.56 ± 0.10
	$\mathbf{d}, \mathbf{m}, S_{\text{LogME}}$	0.55 ± 0.08	0.56 ± 0.09
	$\mathbf{d}, \mathbf{m}, S_{\text{PARC}}$	**0.85** ± 0.04	0.57 ± 0.11
FM (ours)	$\mathbf{d}, \mathbf{m}$	0.54 ± 0.06	**0.57** ± 0.10
	$\mathbf{d}, \mathbf{m}, S_{\text{LFC}}$	0.70 ± 0.12	0.56 ± 0.10
	$\mathbf{d}, \mathbf{m}, S_{\text{LogME}}$	0.55 ± 0.09	0.56 ± 0.10
	$\mathbf{d}, \mathbf{m}, S_{\text{PARC}}$	**0.84** ± 0.05	0.57 ± 0.11

Table 2. We evaluate the average Pearson correlation of predicted performance and the groundtruth performance of 22 models on each benchmark. The ImageNet column is the MS learned with all 409 ImageNet training jobs. The column of LOO (leave-one-out) denotes MS learned with combined training history of ImageNet jobs and all downstream jobs except jobs on the test dataset.

Methods	Features	**19 fine-grained**		**6 DomainNet**		**15 VTAB**	
feature-based MS	S_{LFC} [10]	0.55		0.63		0.14	
	S_{LogME} [60]	0.54		0.52		0.20	
	S_{PARC} [5]	0.54		0.50		0.13	
		ImageNet	LOO	ImageNet	LOO	ImageNet	LOO
LR (ours)	$\mathbf{d}, \mathbf{m}$	0.53	0.66	0.80	0.82	0.29	0.37
	$\mathbf{d}, \mathbf{m}, S_{\text{LFC}}$	**0.67**	**0.74**	**0.84**	**0.85**	**0.38**	**0.41**
	$\mathbf{d}, \mathbf{m}, S_{\text{LogME}}$	0.54	0.65	0.81	0.84	0.30	0.36
	$\mathbf{d}, \mathbf{m}, S_{\text{PARC}}$	0.54	0.66	0.81	0.85	0.30	0.40
FM (ours)	$\mathbf{d}, \mathbf{m}$	0.53	0.65	0.81	0.85	0.35	0.39
	$\mathbf{d}, \mathbf{m}, S_{\text{LFC}}$	**0.64**	**0.74**	0.82	**0.87**	**0.39**	0.41
	$\mathbf{d}, \mathbf{m}, S_{\text{LogME}}$	0.60	0.67	0.82	0.86	0.31	0.40
	$\mathbf{d}, \mathbf{m}, S_{\text{PARC}}$	0.56	0.69	**0.86**	0.86	0.30	**0.43**

4.2.2 Learning from ImageNet training history and evaluating known models on downstream tasks

To evaluate the transferability of learned MS on new datasets, we use all 400+ ImageNet pre-training history as the training data and predict the performance of 22 models on three benchmarks. As shown in the ImageNet columns of Table 2, the learned MS gets comparable or significantly better mean Pearson correlation than feature-based MS on all benchmarks, especially on DomainNet. We see adding extra MS feature can improve performance over learning with only meta features. Note that the Pearson correlation of all MS methods are relatively low on the VTAB benchmark which consists of *structured* and *special* tasks that are much different from ImageNet. It verifies that feature based MS may not transfer well for tasks that are very different from the source task. Since the training data contains 400+ off-the-shelf models are pre-trained on ImageNet-1K or ImageNet-22K, it is prone to learn MS rules such as bigger models lead to better performance, which is mostly true on ImageNet. Because the training data only consists of ImageNet, the lack of dataset diversity leads to the learning of such inductive biases. FM does not show much advantage over LR.

4.2.3 Learning from all training history and evaluating known models on new tasks.

The power of the recommendation formulation is that its performance will improve as more training data is available. For example, if the training history contains models performance on a similar dataset as the target dataset, it is possible that the model works well on the reference dataset will rank higher for the given task. We further increase the number and diversity of training history for learning based methods. For each dataset in a benchmark, we train the MS model with all available training history except the ones for that dataset, i.e., *leave-one-out* (LOO) training data. Table 2 shows that with more training data added, the LOO results improve significantly over the results learned only from ImageNet. Also both LR and FM learned with only meta features (underline) are comparable with the ones trained with additional MS features.

4.3. Ablation Study

Comparing Feature-based MS and Learned MS To understand which datasets benefit from the learned MS, we compare the learned MS models with feature-based MS (e.g.,

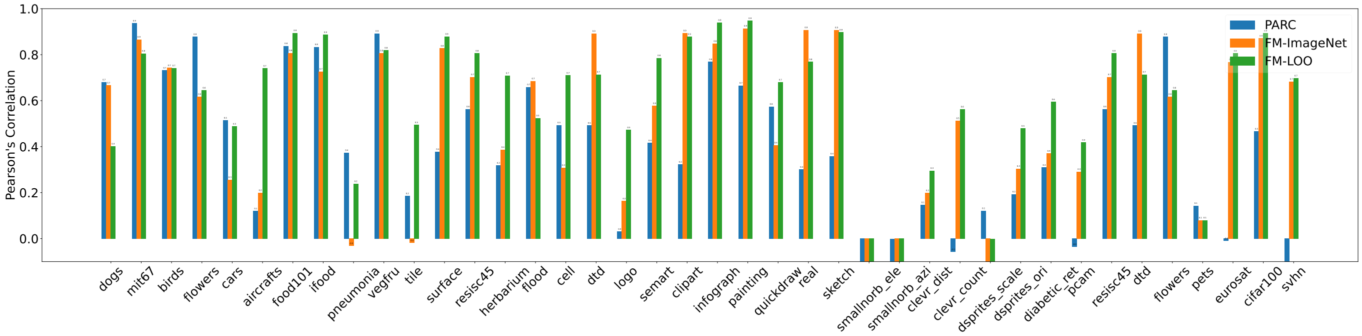


Figure 5. Comparison of learned MS with feature-based MS on each dataset. The FM-ImageNet model is learned with ImageNet only training history and FM-LOO refers to using all datasets except the testing set. Both are trained with only meta feature $\mathbf{d}$ and $\mathbf{m}$.

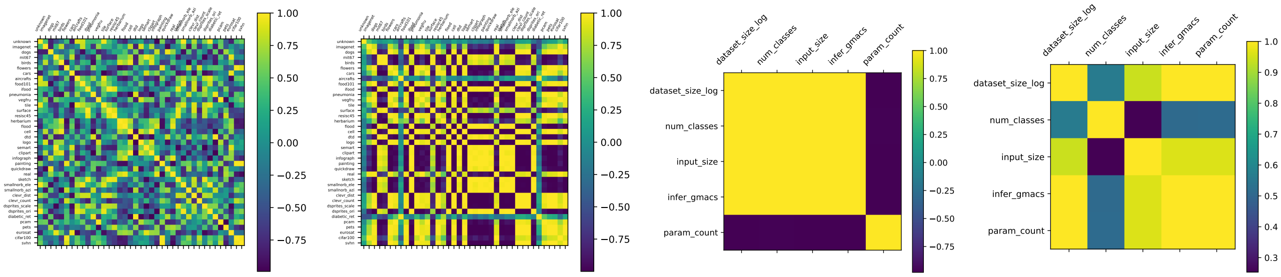


Figure 6. The correlation of selected latent features of FM learned with the ImageNet only training history (a and c) and all training history (b and d). a) and b) select the latent features of dataset IDs, while c) and d) select the scalar features of $\mathbf{d}$ and $\mathbf{m}$.

PARC) on each dataset. As shown in Fig. 5, the learned FM with LOO outperform PARC on 13/19 fine-grained datasets, 6/6 DomainNet datasets, and 10/15 VTAB datasets. The FM model learned with ImageNet only training history transfers well to benchmarks that are similar to ImageNet (e.g, *dogs, mit67, birds, dtd, clipart, real, sketch, cifar100, and svhn*) but underperforms or fails on dissimilar datasets such as *cars, pneumonia, tile*, and *cell*. On the other hand, with more diverse training data, LOO trained model performs significantly better than PARC or ImageNet-only trained model on datasets such as *aircrafts, tile, logo, smart, dsprites_ori, diabetic_ret* and *resisc45*.

Learned Feature Correlations Fig. 6 visualizes the correlation matrix of learned latent representation $\mathbf{u}$ of selected features in FMs trained with different training history. Fig. 6 (a-b) shows the correlation of latent representation of dataset IDs. When only ImageNet history is used, the latent features of other datasets remain random. We see more structured correlation among datasets when more diverse training data is added, i.e., datasets that are similar to each other also have high correlation in their latent representations, such as *dogs* and *pets*. Fig. 6 (c-d) shows clear correlation among scalar features of $\mathbf{d}$ and $\mathbf{m}$ emerges when more data is used, such as dataset size and MACs/parameters, which indicates that larger dataset and larger model weights more. We can also see less correlation among class number and other features. Note that more advanced algorithms such as field-aware FM (FFM) [26] may further improve the performance, in which the correlation of features belonging to the same filed (e.g., dataset) are not learned.

5. Related Work

Model Selection MS methods can be categorized based on whether the source dataset is available. When source data is available, models are in the same architecture and differ only in pre-trained domains, the features and labels of source data and target data can be compared with methods such as EMD [9] and NCE [54]. Probabilistic based methods such as H-Score [2], LEEP [42], $\mathcal{N}$LEEP [35] and LogME [60] estimate the likelihood or the marginalized likelihood of labeled target examples, assuming that a linear classifier is added on top of the pre-trained model. Recent TransRate [22] measures the mutual information between the backbone features and the labels, and also extends to layer selection. LFC [10] approximates the fine-tuning dynamics by looking at a linearization of the source model around the pre-trained weights with the assumption that fine-tuned weights tend to remain close to the pre-trained weights. PARC [5] main differs with LFC with the choice of correlation metric. Note that an improved PARC adds model depth with heuristic weight, which is essentially a linear combination of MS score with model's meta feature.

Learning to Recommend There are also learning based methods to recommend dataset, hyperparameters and techniques for a given task. Neural Data Server [59] provides a search engine to find the most useful transfer learning data for the target domain. MS can also be viewed as a hyperparameter selection problem. HyperStar [40] learns to predict the performance of a hyperparameter set for a given image classification task with a end-to-end trained CNN. The work [14] is most relevant to us, in which a general probabilistic model matrix factorization is learned for ML pipeline selection. A learning based approach for MS is [46], which introduced a model routing algorithm for a large number of expert models. Its domain prediction method classifies the expert from the image via an auxiliary network, which is a classification-based approach as we mentioned in Sec 3.2.

6. Conclusion

The nature of long-tailed tasks determines that no single model works best for all tasks, which makes model selection in a model zoo with diverse inductive biases necessary. Instead of manually designing MS criteria, learning the relationship between tasks and models via recommendation models can be more efficient, effective and scalable to new meta features and models, and it can be continuously improved with the growing volume of training history. This makes the framework applicable to other selection problems as well such as selecting optimal models and hyperparameters at the same time.

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. 1

[2] Yajie Bao, Yang Li, Shao-Lun Huang, Lin Zhang, Amir R Zamir, and Leonidas J Guibas. An information-theoretic metric of transferability for task transfer learning. In *ICIP*, 2019. 8

[3] Björn Barz, Kai Schröter, Moritz Münch, Bin Yang, Andrea Unger, Doris Dransch, and Joachim Denzler. Enhancing flood impact analysis using interactive retrieval of social media images. *arXiv preprint arXiv:1908.03361*, 2019. 11

[4] Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. Material recognition in the wild with the materials in context database. 2015. 11

[5] Daniel Bolya, Rohit Mittapalli, and Judy Hoffman. Scalable diverse model selection for accessible transfer learning. In *NeurIPS*, 2021. 2, 3, 4, 6, 7, 8

[6] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *ECCV*, 2014. 11, 16

[7] Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017. 11

[8] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *CVPR*, 2014. 11

[9] Yin Cui, Yang Song, Chen Sun, Andrew Howard, and Serge Belongie. Large scale fine-grained categorization and domain-specific transfer learning. In *CVPR*, 2018. 8

[10] 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 preprint arXiv:2102.00084*, 2021. 1, 2, 3, 5, 6, 7, 8

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 6, 12

[12] Kshitij Dwivedi and Gemma Roig. Representation similarity analysis for efficient task taxonomy & transfer learning. In *CVPR*, 2019. 1

[13] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *CVPR Workshop*, 2004. 11

[14] Nicolo Fusi, Rishit Sheth, and Melih Elibol. Probabilistic matrix factorization for automated machine learning. *NeurIPS*, 31, 2018. 8

[15] Noa Garcia and George Vogiatzis. How to read paintings: semantic art understanding with multi-modal retrieval. In *ECCV Workshops*, 2018. 11

[16] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013. 11

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 6, 12

[18] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2019. 11

[19] Saihui Hou, Yushan Feng, and Zilei Wang. Vegfru: A domain-specific dataset for fine-grained visual categorization. In *ICCV*, 2017. 11

[20] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 6, 12

[21] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017. 6, 12

[22] Long-Kai Huang, Junzhou Huang, Yu Rong, Qiang Yang, and Ying Wei. Frustratingly easy transferability estimation. In *ICML*, 2022. 8

[23] Yibin Huang, Congying Qiu, and Kui Yuan. Surface defect saliency of magnetic tile. *The Visual Computer*, 36(1):85–96, 2020. 11

[24] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*, 2017. 11

[25] Alexis Joly and Olivier Buisson. Logo retrieval with a contrario visual query expansion. In *ACM MM*, pages 581–584, 2009. 11

[26] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. Field-aware factorization machines for ctr prediction. In *Proceedings of the 10th ACM conference on recommender systems*, 2016. 8

[27] Kaggle and EyePacs. Kaggle diabetic retinopathy detection, jul 2015. 11

[28] Parneet Kaur, , Karan Sikka, Weijun Wang, serge Belongie, and Ajay Divakaran. Foodx-251: A dataset for fine-grained food classification. *arXiv preprint arXiv:1907.06167*, 2019. 11

[29] Daniel S Kermany, Michael Goldbaum, Wenjia Cai, Carolina CS Valentim, Huiying Liang, Sally L Baxter, Alex McKeown, Ge Yang, Xiaokang Wu, Fangbing Yan, et al. Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell*, 172(5):1122–1131, 2018. 11

[30] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, CVPR*, 2011. 3, 11

[31] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *CVPR*, 2019. 3, 5

[32] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013. 11

[33] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 11

[34] Yann LeCun, Fu Jie Huang, and Léon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *CVPR*, 2004. 11

[35] Yandong Li, Xuhui Jia, Ruoxin Sang, Yukun Zhu, Bradley Green, Liqiang Wang, and Boqing Gong. Ranking neural checkpoints. In *CVPR*, 2021. 8

[36] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021. 6, 12, 17

[37] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *arXiv preprint arXiv:2201.03545*, 2022. 6, 12

[38] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. Technical report, 2013. 3, 11

[39] Loic Matthey, Irina Higgins, Demis Hassabis, and Alexander Lerchner. dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/, 2017. 11

[40] Gaurav Mittal, Chang Liu, Nikolaos Karianakis, Victor Fragoso, Mei Chen, and Yun Fu. Hyperstar: Task-aware hyperparameters for deep networks. In *CVPR*, 2020. 8

[41] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011. 11

[42] Cuong V Nguyen, Tal Hassner, Cedric Archambeau, and Matthias Seeger. Leep: A new measure to evaluate transferability of learned representations. *arXiv preprint arXiv:2002.12462*, 2020. 8

[43] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics & Image Processing*. IEEE, 2008. 11

[44] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *CVPR*, 2012. 11

[45] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *ICCV*, 2019. 5, 11

[46] Joan Puigcerver, Carlos Riquelme, Basil Mustafa, Cedric Renggli, André Susano Pinto, Sylvain Gelly, Daniel Keysers, and Neil Houlsby. Scalable transfer learning with expert models. In *ICLR*, 2021. 4, 8

[47] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 1

[48] Sivaramakrishnan Rajaraman, Sameer K Antani, Mahdieh Poostchi, Kamolrat Silamut, Md A Hossain, Richard J Maude, Stefan Jaeger, and George R Thoma. Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images. *PeerJ*, 6:e4568, 2018. 11

[49] Steffen Rendle. Factorization machines. In *ICDM*, 2010. 4

[50] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *CVPR workshops*, pages 806–813, 2014. 11

[51] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 12

[52] Kiat Chuan Tan, Yulong Liu, Barbara Ambrose, Melissa Tulig, and Serge Belongie. The herbarium challenge 2019 dataset. *arXiv preprint arXiv:1906.05372*, 2019. 11

[53] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019. 6, 12, 16

[54] Anh T Tran, Cuong V Nguyen, and Tal Hassner. Transferability and hardness of supervised classification tasks. In *ICCV*, 2019. 8

[55] Bastiaan S Veeling, Jasper Linmans, Jim Winkens, Taco Cohen, and Max Welling. Rotation equivariant CNNs for digital pathology. June 2018. 11

[56] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 11, 16

[57] Peng Wang, An Yang, Rui Men, Junyang Lin, Shuai Bai, Zhikang Li, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework. *arXiv preprint arXiv:2202.03052*, 2022. 1

[58] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, pages 3485–3492, June 2010. 11

[59] Xi Yan, David Acuna, and Sanja Fidler. Neural data server: A large-scale search engine for transfer learning data. In *CVPR*, 2020. 8

[60] Kaichao You, Yong Liu, Jianmin Wang, and Mingsheng Long. Logme: Practical assessment of pre-trained models for transfer learning. In *ICML*, 2021. 2, 3, 6, 7, 8

[61] Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, et al. A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv preprint arXiv:1910.04867*, 2019. 5, 11, 17

[62] Chongzhi Zhang, Mingyuan Zhang, Shanghang Zhang, Daisheng Jin, Qiang Zhou, Zhongang Cai, Haiyu Zhao, Shuai Yi, Xianglong Liu, and Ziwei Liu. Delving deep into the generalization of vision transformers under distribution shifts. *arXiv preprint arXiv:2106.07617*, 2021. 17

A. Datasets, Models and Fine-tuning Hyperparameters

Datasets The **19 Fine-grained datasets** contain 19 commonly used fine-grained visual classification datasets, covering a wide range of domains, including *objects*, *scene*, *plants*, *animals*, *food*, *texture*, *medical*, *logo* and *art*. The **DomainNet** [45] benchmark is designed for evaluating multi-source domain adaptation in object recognition. It contains 0.6 million images across 6 domains (*clipart*, *infograph*, *painting*, *quickdraw*, *real*, and *sketch*). All domains include 345 categories (classes) of objects. We use the official train/test splits in our experiments. The **VTAB** [61] benchmark is designed for evaluating the transferability of pre-trained models. It consists of 19 datasets and the tasks are categorized into *natural*, *structured* and *special*. Some of the datasets can also be categorized as in the 19 fine-grained datasets. Note that there are some datasets also exists in the 19 fine-grained datasets. We include them when reporting the VTAB performance. Currently we used 15 of the 19 datasets for VTAB, covering all the categories. More detailed information of each dataset can be found in Table 3.

Table 3. Datasets statistics. For the *aircrafts, flowers* and *surface* dataset, the original training set and validation set are combined following the practice. Note datasets noted with * are not included in our experiments.

Benchmark	Dataset Names	Alias	Domain	Classes	Training	Test
19 Fine-grained	Stanford Dogs [30]	dogs	animals	120	12,000	8,580
	CUB-Birds 200 [56]	birds	animals	200	5,994	5,794
	Oxford Flowers [43]	flowers	plants	102	2,040	6,149
	VegFru [19]	vegfru	plants	290	29,000	116,156
	Herbarium 2019 [52]	herbarium	plants	683	31, 546	2,679
	FGVC Aircrafts [38]	aircrafts	objects	100	6,667	3,333
	Stanford Cars [32]	cars	objects	196	8,144	8,041
	MIT Indoor-67 [50]	mit67	scene	67	5,360	1,340
	European Flood Depth [3]	flood	scene	2	3153	557
	NWPU Resisc45 [7]	resisc45	scene	45	25,200	6,300
	Food-101 [6]	food101	food	101	12,000	8,500
	iFood [28]	ifood	food	251	118,475	11,994
	Describable Textures [8]	dtd	texture	47	4,230	1,410
	Open Surface-2500 [4]	surface	texture	23	48,875	8,625
	Magnetic Tile [23]	tile	texture	5	1,008	336
	Pneumonia [29]	pneumonia	medical	2	25,216	624
	Malaria Cell Images [48]	cell	medical	2	20,668	6,890
	BelgaLogos [25]	logo	logo	27	7,500	2,500
	SemArt [15]	semart	art	26	18,174	3,208
DomainNet	Clipart	clipart	clipart	345	33,525	14,604
	Real	real	real	345	120,906	52,041
	Quickdraw	quickdraw	quickdraw	345	120,750	51,750
	Painting	painting	painting	345	50,416	21,850
	Inforgraph	inforgraph	inforgraph	345	36,023	15,582
	Sketch	sketch	sketch	345	48,212	20,916
VTAB	Caltech101* [13]	caltech101	natural - objects	101	3,060	6,084
	SUN397* [58]	sun397	natural - scene	397	73,257	26,032
	Oxford Flowers [43]	flowers	natural - plants	102	2,040	6,149
	CIFAR-100 [33]	cifar100	natural - objects	100	50,000	10,000
	SVHN [41]	svhn	natural - object	10	73,257	26,032
	Oxford IIIT Pet [44]	pets	natural - animal	37	3,680	3,669
	Describable Textures [8]	dtd	natural - texture	47	4,230	1,410
	NWPU Resisc45 [7]	resisc45	specialized - scene	45	25,200	6,300
	EuroSAT [18]	eurosat	specialized - scene	10	20,250	6,750
	Diabetic Rethinopathy [27]	retinopathy	specialized - medical	5	35,126	53,576
	PatchCamelyon [55]	pcam	specialized - medical	2	262,145	32,769
	CLEVR distance [24]	clevr_dist	structured	7	70,000	15,000
	CLEVR counting [24]	clevr_dist	structured	8	70,000	15,000
	Dmlab Frames*	dmlab	structured	6	65,550	22,628
	dSprites orientation [39]	dsprites_ori	structured	40	663,552	73,728
	dSprites location [39]	desprites_loc	structured	6	663,552	73,728
	KITTI distance [16]*	kitti_dist	structured	4	7,481	7,518
	Small NORB azimuth [34]	smallnorb_elevation	structured	18	24,300	24,300
	Small NORB elevation [34]	smallnorb_azimuth	structured	9	24,300	24,300

Models The `TIMM` and `torchvision` model zoo collected over 590 ImageNet pre-trained models in various architectures and training recipes. We filtered out 409 models that can be fine-tuned with batch size 32 and evaluated the single image inference latency of the 400+ models on single GPU. The scatter plot of latency and accuracy can be seen in Fig. 7(a). We can identify the Pareto frontier models of the 400+ models, spanning the latency from 3 ms to 120 ms. We select 22 widely used models that are near the Pareto Front curve, which covers a wide range of architecture families, including ReseNet [17], DenseNet [21], MobileNet [20], EfficientNet [53], ViTs [11], Swin-T [36] and ConvNeXt [37]. The detailed statistics of the selected 22 models can be seen in Table 4.

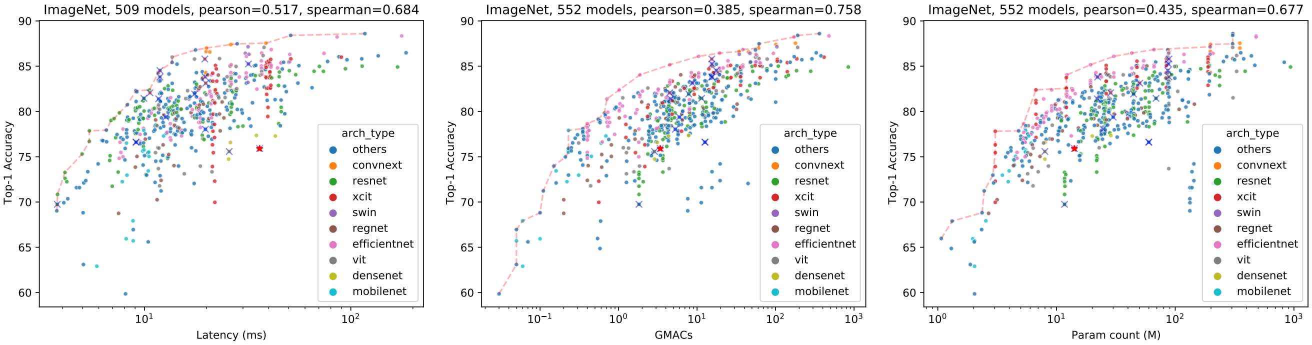


Figure 7. The statistics of the 500+ ImageNet pre-trained models. The latency is measured on V100 GPU with batch size 1. The dashed line connects the Pareto frontier models. The blue crossed models are our selected 22 models and the red crossed dot is the reference model - DenseNet-169.

Table 4. The statistics of the 22 models, which are ranked by their single image inference latency (ms) on the V100 GPU.

	Model Name	Arch Family	Acc	Pretrain	Img Size	Latency	FLOPs	#Params
1	resnet18	resnet	69.74	IN-1K	224	3.78	1.82	11.69
2	mobilenet_v2	mobilenet	71.88	IN-1K	224	7.33	0.31	3.50
3	mixer_b16_224	others	76.61	IN-1K	224	9.10	12.62	59.88
4	mixer_b16_224_in21k	others	-	IN-21K	224	9.10	12.62	59.88
5	wide_resnet50_2	resnet	81.45	IN-1K	224	9.94	11.43	68.88
6	convnext_tiny	convnext	82.06	IN-1K	224	10.63	4.47	28.59
7	vit_small_patch16_224	vit	81.40	IN-1K	224	11.71	4.61	22.05
8	vit_small_patch16_384	vit	83.81	IN-1K	384	11.88	15.52	22.2
9	vit_base_patch16_224	vit	84.53	IN-1K	224	11.88	17.58	86.57
10	vit_base_patch16_224_in22k	vit	-	IN-21K	224	11.88	17.58	86.57
11	resmlp_24_224	others	79.38	IN-1K	224	12.67	5.96	30.02
12	efficientnet_b0	efficientnet	76.30	IN-1K	224	15.06	0.40	5.29
13	resnet101	resnet	81.93	IN-1K	224	17.48	7.83	44.55
14	convnext_base	convnext	83.82	IN-1K	224	19.68	15.38	88.59
15	convnext_base_in22ft1k	convnext	85.80	IN-21K-1K	224	19.60	14.38	88.59
16	gmixer_24_224	others	78.04	IN-1K	224	19.74	5.28	24.72
17	convnext_small	convnext	83.13	IN-1K	224	19.80	8.70	50.22
18	efficientnet_b3	efficientnet	81.10	IN-1K	300	24.27	2.01	12.23
19	densenet121	densenet	75.58	IN-1K	224	25.73	2.87	7.98
20	swin_base_patch4_window7_224	swin	85.25	IN-1K	224	31.90	15.47	87.77
21	swin_base_patch4_window7_224_in22k	swin	-	IN-21K	224	31.90	15.47	87.77
22	densenet169	densenet	75.90	IN-1K	224	36.13	3.40	14.15

Fine-tuning Hyperparameters All models are trained with a single GPU with the same settings with the hyperparameter search ranges. We performed fine-tuning with following hyper-parameters: we fine-tune 30 epochs with SGD with Nesterov momentum 0.9, batch size of 32 and weight decay of 10^{-4}. The learning rate η decays by $0.1\times$ at 15th and 25th epochs. We performed a grid search of with various initial learning rates and data augmentation strategies, i.e., $\eta_0 \in \{0.05, 0.01, 0.005, 0.001\}$ and data_aug $\in \{\texttt{rrcrop}, \texttt{rcrop}\}$. Here `rrcrop` stands for random resized cropping, which randomly crops ratio ranging from 0.08 to 1.0 with random aspect ratio between [3/4, 4/3], which is adopted in [51]. And `rcrop` stands for random cropping, which differs with with `rrcrop` in that it uses fixed cropping ratio (0.875). We report the best top-1 test accuracy of the 8 trials.

B. Feature-Based Model Selection

More Model Selection Failure Cases In additional to the model selection results on the 19 fine-grained datasets (Fig. 2 in Sec. 2.2, here we show the MS results on DomainNet and VTAB in Fig. 8. The feature-based MS methods perform relatively well on DomainNet datasets (*clipart*, *infograph*, *painting*, *quickdraw*, *real*, and *sketch*). However, the MS methods have weak or even negative correlations for some of the VTAB tasks. For example, the structured tasks including *smallnorb*, *clevr* and *dsprites*. Even the simple task SVHN has negative correlations, which indicates that the feature-based MS methods can fail to estimate the relative performance with only feature information.

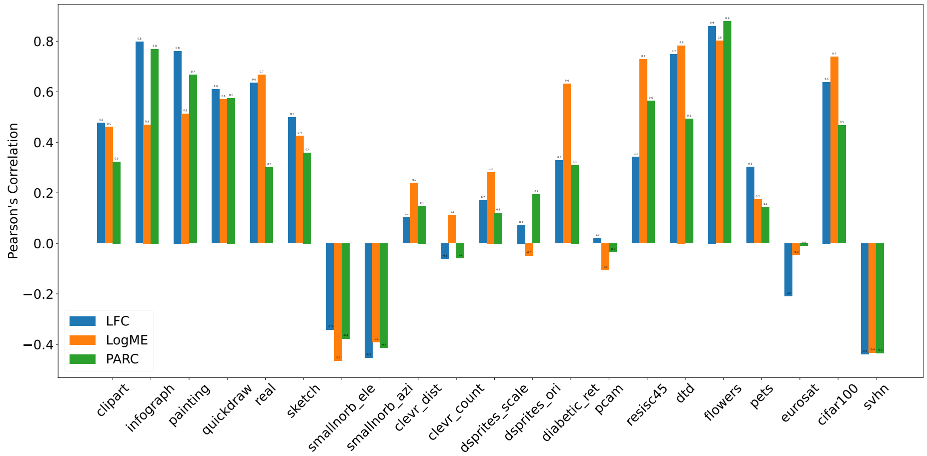


Figure 8. Comparison of MS methods on DomainNet and VTAB. The first 6 datasets (*clipart*, *infograph*, *painting*, *quickdraw*, *real*, and *sketch*) are from DomainNet, and the rest of the datasets are from VTAB.

Dataset Sampling For feature-based MS on ImageNet, we sample 2,000 images from the ImageNet training set, with the number of images per class set to 2. We use the fixed sub-sampled prob set for evaluating all models. For MS on downstream tasks, we sample at most 2,000 images with the constraint that no class has more than 25 images.

C. Model Recommendation

Feature Embedding Table 5 lists the details of the features used in learning based MS. The categorical features are converted to one-hot vectors. And all features are normalized with their minimum and maximum values across all datasets, so that the maximums and minimum values are 1 and 0 after normalization. The visualization of the real embedding of all available fine-tuning tasks can be seen in Fig 9.

Training Details We implement the LR and FM algorithms with the `xlearn` library. The regression models with MAE loss are trained with SGD until the loss converges. The initial learning rate is 0.2 and the regularization λ is 0.002. Instance-wise normalization is disabled.

D. Heterogeneous Model Zoo and Architecture Bias

The existence of inductive bias for different architectures indicates that there is no single best architecture for all tasks with different characteristics. Our hypothesis is that the optimal architecture for transfer learning is task dependent. To verify this, we perform experiments with diverse datasets and architectures. We identify the existence of architecture bias for different datasets, which confirms the need of task dependent architecture selection. Fig 10 shows the fine-tuning performance of the 22 models on 40 downstream tasks, from which we can see the ranking of the model for each task can be significantly different. It also demonstrates that the best performing model can be task dependent.

Table 5. The complete features for embedding fine-tuning tasks for learning-based MS.

Field idx	Field Name	Feature Name	Type	One-hot	Log	Dimension	Min	Max
1	dataset	dataset id	category	Yes	No	41	0	40
1	dataset	dataset size	scalar	No	Yes	1	1008	1200000
1	dataset	number of classes	scalar	No	Yes	1	2	1000
2	model	architecture id	category	Yes	No	500	0	409
2	model	architecture family id	category	Yes	No	10	0	9
2	model	pre-trained dataset id	category	Yes	No	3	0	2
2	model	input size	scalar	No	Yes	1	106	448
2	model	GMACs (G)	scalar	No	Yes	1	0.03	46.95
2	model	#Parameters (G)	scalar	No	Yes	1	1.88	88.59
3	MS score	LFC	scalar	No	No	1	0.002	0.792
3	MS score	LogME	scalar	No	No	1	-0.905	2.209
3	MS score	PARC	scalar	No	No	1	0.085	80.358

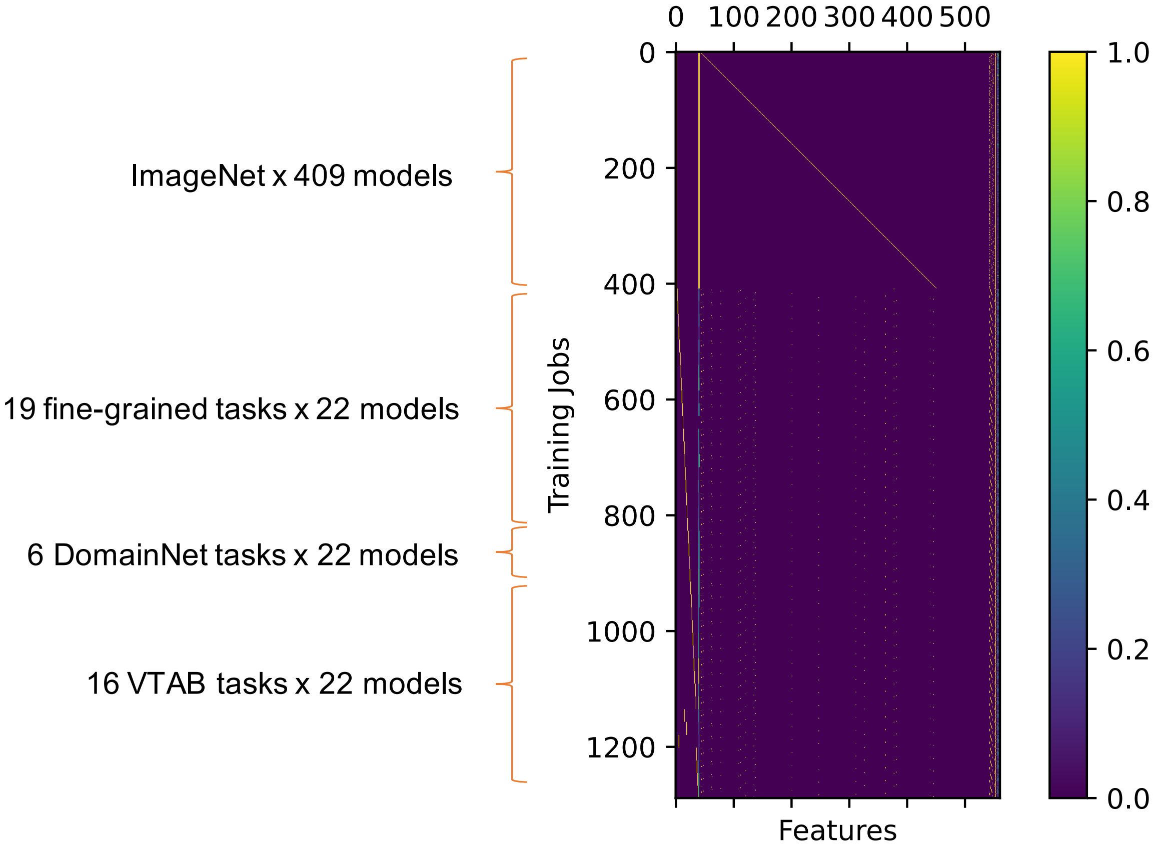


Figure 9. Visualization of the normalized feature embedding of all training jobs, including 400+ ImageNet training jobs, 418 training jobs on the 19 fine-grained tasks, 132 training jobs on 6 DomainNet datasets and 352 training jobs on the VTAB datasets.

D.1. Architecture bias

Given a task and a set of well pre-trained models in different architectures, we say there is an architecture bias for a task if one architecture obtains the best performance and outperforms the second best architecture by a large margin (e.g., > 2% top-1 accuracy). To justify the significance of architecture bias, we show the following facts for each benchmark: a) the performance distribution of each architecture over a baseline model across all downstream tasks. b) the performance gain of the best model over the second best performing model for each task. Fig. 11 ranks the models by their mean performance, from which we can

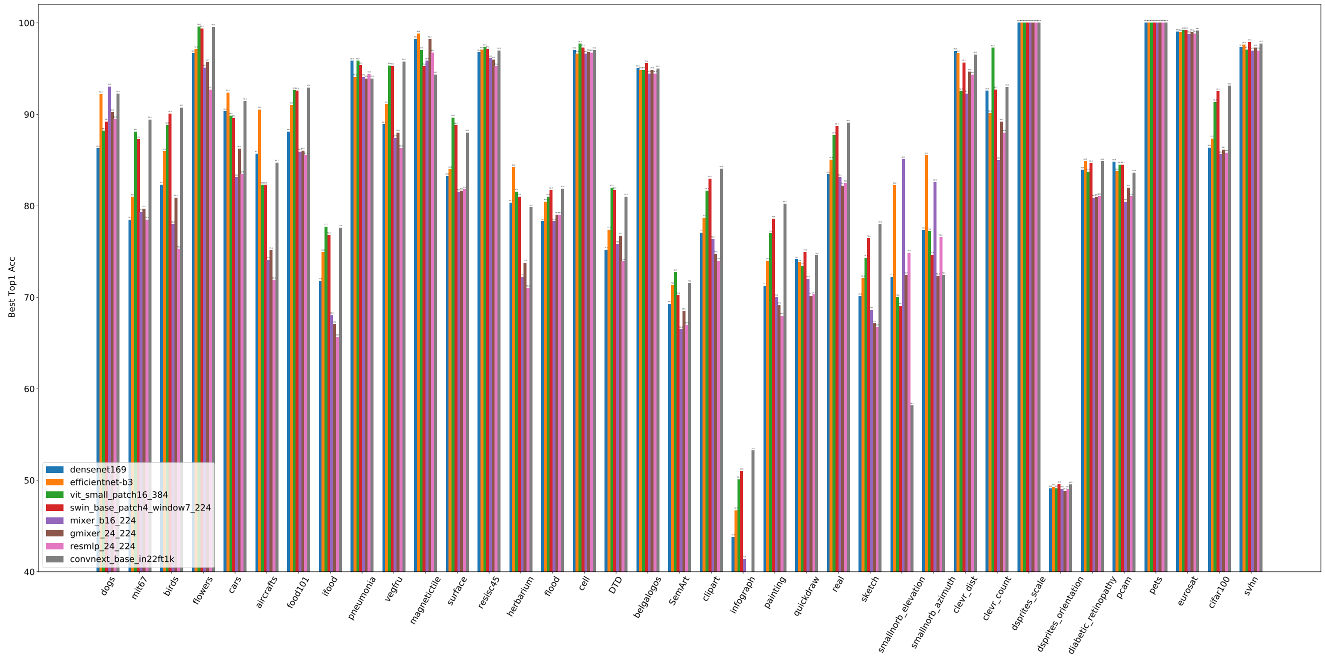


Figure 10. The performance of different architectures on the 19 fine-grained datasets, DomainNet and VTAB. The best performing architecture for different tasks can be different, For example, EfficientNet-B3 with higher resolution input performs best on *aircrafts*, *herbarium* and *smallnorb_azimuth*; Swin-B/16 wins on *surface*, *texture (DTD)* and *semart*.

see ConvNeXt, ViTs, Swin-T, EfficientNet are top ranked models in terms of average performance gain over DenseNet-169. Note that although ConvNeXt has the strongest average performance, it is not always the best for all tasks. Other architectures can outperform ConvNeXt significantly for datasets like *aircrafts*, *magnetictile*, *herbarium*, *dogs*, indicating the existence of architecture bias for those datasets. Similarly, we observe stronger architecture bias on structured tasks in *VTAB*, such as *smallnorb elevation* and *clevr count*. For *DomainNet*, we find ConvNeXt with ImageNet-22K pre-training performs best on 5 out of 6 domains (e.g., *clipart*, *inforgraph*, *painting*) with significant performance gains over DenseNet169 (> 6%). However, the best performing architecture on *quickdraw* is Swin-T and the differences among the architectures are small (<1%) (Fig. 12c).

Table 6. The Wilcoxon signed-rank test on whether the row model is significantly better than the column model on the 19 fine-grained tasks. The table shows the p-value. The bold values indicate that the row model is statistically better than the column model ($p < 0.05$). No model is statistically better than all other models.

	ViT-S/16-384	Swin-B-P4-W7-in22k	Swin-B-P4-W7	Efficientnet-B3	ViT-B/16-224-in21k
ConvNeXt-B-in22ft1k	0.492	0.384	0.198	0.084	**0.003**
ViT-S/16-384	-	0.147	0.072	0.107	**0.002**
Swin-B-P4-W7-in22k	0.862	-	0.098	0.121	**0.003**
Swin-B-P4-W7	0.928	0.909	-	0.156	**0.016**
Efficientnet-B3	0.893	0.887	0.853	-	0.779

Statistical Test We have empirically verified the hypothesis that the optimal architecture for transfer learning is task dependent and there is no single best model that performs best on every datasets. Here we performed statistical tests for the hypothesis. We conduct non-parametric paired one-tailed t-test (the Wilcoxon signed-rank test) on whether the selected model's performance is greater than other fine-tuning methods across 19 fine-tuning tasks. The null hypothesis H_0 states that the mean performance difference between selected model and baseline model is zero. The alternative hypothesis H_1 states that the selected model outperforms the baseline model. We pick the top 6 models with the best average performance on the

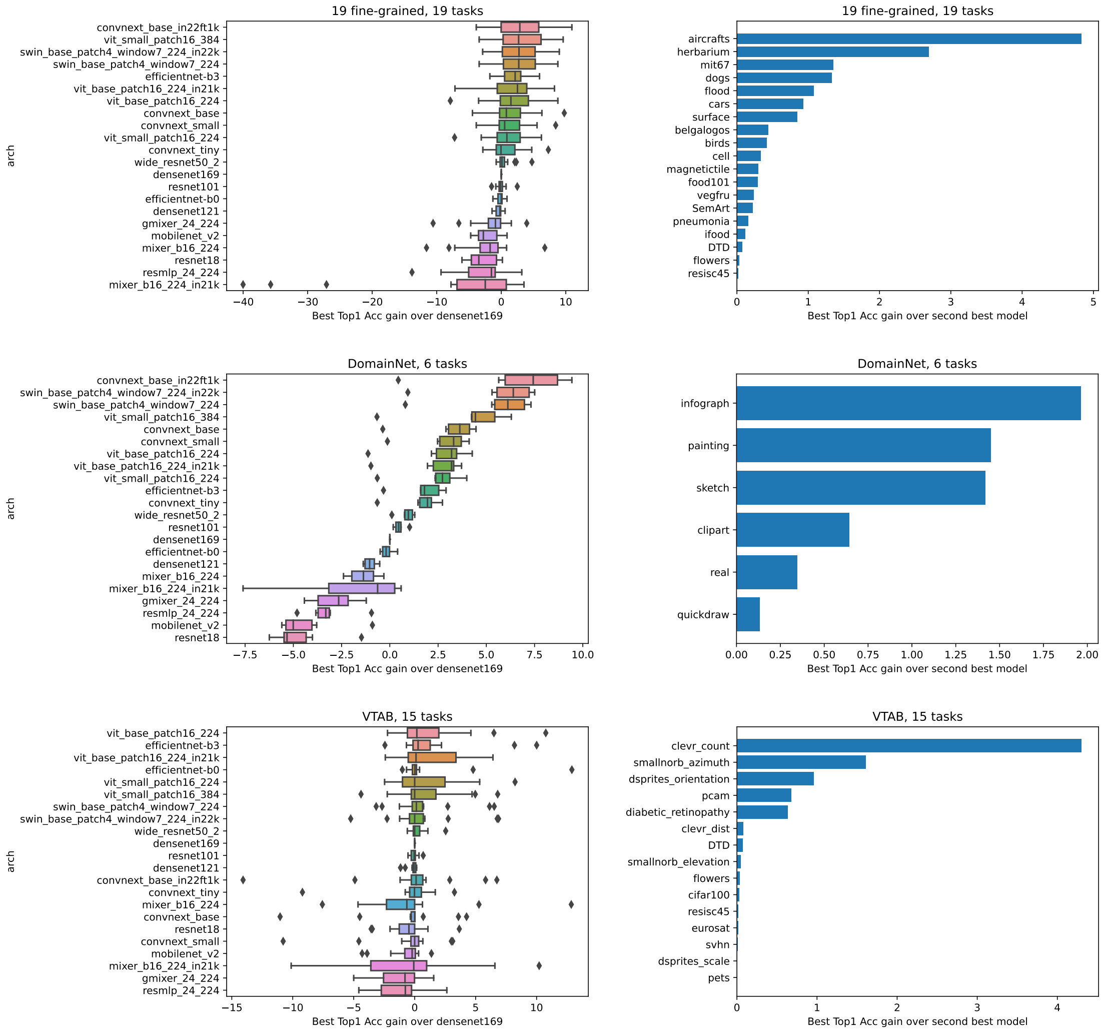


Figure 11. The significance of architecture bias on the three benchmarks. The first column shows the performance gain over DenseNet-169 on downstream tasks for each model. The models are ranked by their mean accuracy gain. The second column shows the performance gain of the optimal model over the second best performing model for each dataset.

19 fine-grained tasks (Fig. 11) and check whether any of them can be significantly better than others. Table 6 presents the p-values of each test, with the number of observations equal to 19 for each model compared. There is no single model that outperforms all other models.

D.2. Why do certain models work well on certain datasets?

We investigate the reason why certain models perform better on certain datasets than others. As shown in Fig. 11, the top 4 best performing models are ConvNeXt-B-in22ft1k, Efficientnet-B3, ViT-S/16-384 and Swin-B-in22k. Here we analyze why they are chosen for certain tasks and what distinguish them from other models.

- ConvNeXt-B-in22ft1k performs best on many downstream tasks, such as *birds* [56] and *food101* [6]. One reason is that this model is obtained with strong pre-training on ImageNet-22k and then fine-tuned on ImageNet-1K, giving a bias towards datasets that are close to ImageNet.

- Efficientnet-B3 [53] is chosen over ConvNeXt for *aircrafts*. Note that EfficientNet-B3 adopts a *higher resolution* for input images (300 instead of 224). Dataset such as *aircrafts* benefits from the high resolution to make the subtle differences

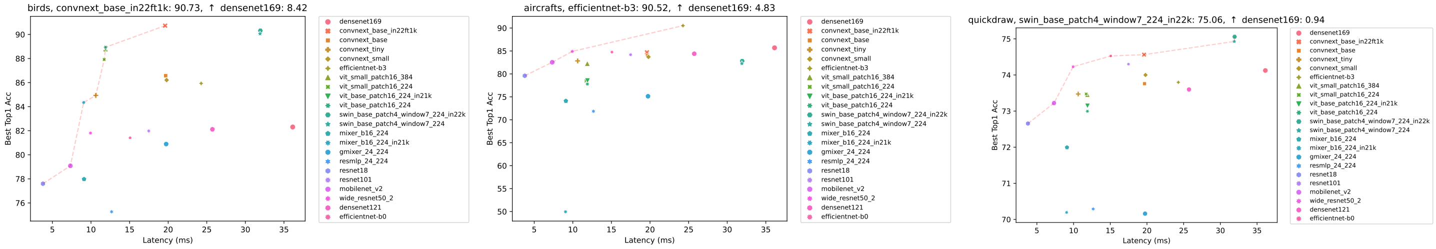


Figure 12. The Pareto front models for *birds*, *aircrafts* and *DomainNet-quickdraw*. ConvNeXt, EfficientNet-B3 and Swin-T are the best performing models and have large margin over the other architectures.

noticeable. EfficientNet-B3 also has significant better performance on structured tasks such as *smallnorb* and *clevr* in VTAB, which are synthetic 3D objects tasks such as counting and angle estimation. Similar observation is also made in [61] that *structured* tasks behaves differently with nature images.

- Swin-B [36] performs best on *quickdraw* (Fig. 12(c)), which has no color or texture information but only shapes. It suggests that Swin-T has the advantage of capturing the structure information. However the task is so simple the architecture bias or pre-training makes not too much on performance difference (the difference between ResNet-18 and the wining Swin-T is only 2%). Similarly previous work [62] finds that ViTs are better than CNNs on this task, and conclude that ViTs are better preserving shape and structure information.

D.3. More Pareto front results

We have shown that the optimal model is dataset dependent. A natural question to ask is that whether the Pareto front models for ImageNet continue to be on the Pareto frontier for other downstream tasks. Similar to Fig. 7(a), we plot the scatter plot of latency and performance for three datasests in Fig 12 and show more results in Fig. 13. We can see that the Pareto frontier models are actually task dependent, which suggests the need to perform dataset dependent search.

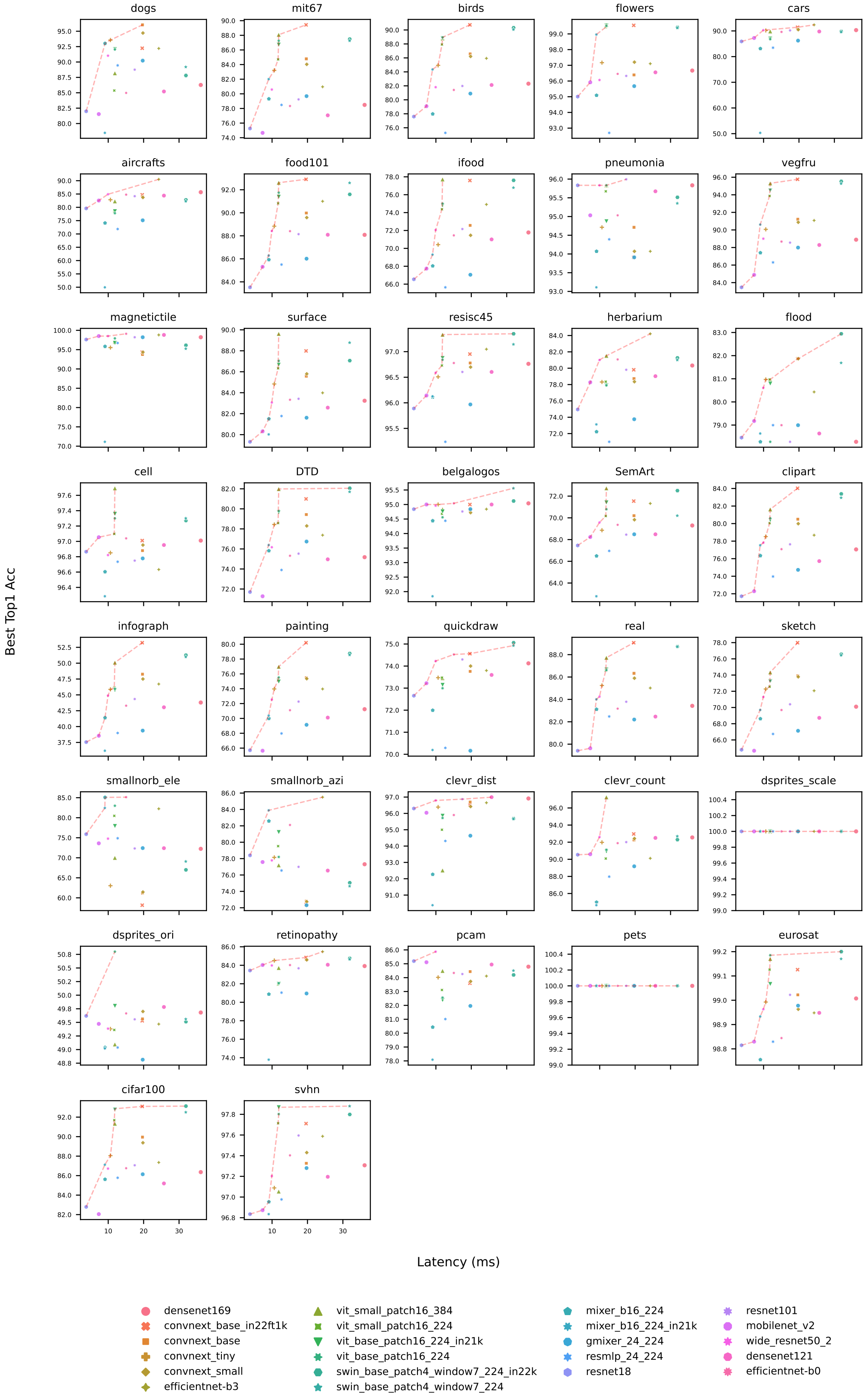


Figure 13. The Pareto front models can be task dependent.