Which Model to Transfer? Finding the Needle in the Growing Haystack

Cedric Renggli[*]   André Susano Pinto   Luka Rimanic   Joan Puigcerver
ETH Zurich   Google Research   ETH Zurich   Google Research

Carlos Riquelme   Ce Zhang   Mario Lucic
Google Research   ETH Zurich   Google Research

Abstract

Transfer learning has been recently popularized as a data-efficient alternative to training models from scratch, in particular for computer vision tasks where it provides a remarkably solid baseline. The emergence of rich model repositories, such as TensorFlow Hub, enables the practitioners and researchers to unleash the potential of these models across a wide range of downstream tasks. As these repositories keep growing exponentially, efficiently selecting a good model for the task at hand becomes paramount. We provide a formalization of this problem through a familiar notion of regret *and introduce the predominant strategies, namely* task-agnostic *(e.g. ranking models by their ImageNet performance) and* task-aware *search strategies (such as* linear *or* kNN *evaluation). We conduct a large-scale empirical study and show that both task-agnostic and task-aware methods can yield high regret. We then propose a simple and computationally efficient hybrid search strategy which outperforms the existing approaches. We highlight the practical benefits of the proposed solution on a set of 19 diverse vision tasks.*

1. Introduction

Services such as TensorFlow Hub or PyTorch Hub[1] offer a plethora of pre-trained models that often achieve state-of-the-art performance on specific tasks in the vision domain. The predominant approach, namely choosing a pre-trained model and *fine-tuning* it to the downstream task, is an effective and data efficient approach [12, 13, 18, 23, 33, 35]. Perhaps surprisingly, this approach is also effective when the pre-training task is significantly different from the target task, such as when applying an ImageNet pre-trained model to diabetic retinopathy classification [20]. Fine-tuning often entails adding several more layers to the pre-trained deep network and tuning all the parameters using a limited amount of downstream data. Due to the fact that all parameters are being updated, this process can be extremely costly in terms of compute [35]. Fine-tuning all models to find the best performing one is becoming computationally infeasible. A more efficient alternative is to simply train a cheap classifier on top of the learned representation (e.g. pre-logits). However, the performance gap with respect to fine-tuning can be rather large [13, 14].

This raises a very practical question: *Given a new task, how to pick the best model to fine-tune?* This question was intensively studied in recent years and existing approaches can be divided into two groups: (a) *task-agnostic* model search strategies, which rank pre-trained models independently of the downstream task (e.g. ranking the models by ImageNet accuracy, if available) [14], and (b) *task-aware* model search strategies, which make use of the downstream dataset in order to rank models (e.g. *k*NN classifier accuracy as a proxy for fine-tuning accuracy, or using the meta-learned Task2Vec representations) [1, 16, 22]. Most of the prior work attempts to answer this question by using a homogeneous sets of pre-trained models whereby the models share the same architecture or they were trained on the same dataset. However, this does not reflect the current landscape of models available in online repositories.

As a result, several practically relevant questions remained open. Firstly, how do existing methods compare in the presence of both "generalist models" (e.g. models trained on a relatively diverse distribution such as ImageNet) and "expert models" (e.g. trained on domain-specific datasets such as plants), across a diverse collection of datasets? Secondly, *is there a method which strikes a good balance between computational cost and performance?*

Our contributions. In this paper, we provide a large-scale, systematic empirical study of these questions. (i) We define and motivate the model search problem through a notion of regret. We conduct the first study of this problem in a realistic setting focusing on heterogeneous model pools. (ii) We consider 19 downstream tasks on a heterogeneous set of 46 models grouped into 5 representative sets. (iii) We highlight the dependence of the performance of each strat-

---

[*]Work done while interning at Google Research. Correspondence to C. Renggli (cedric.renggli@inf.ethz.ch) and M. Lucic (lucic@google.com).
[1]https://tfhub.dev and https://pytorch.org/hub

egy on the constraints of the model pool, and show that, perhaps surprisingly, both task-aware and task-agnostic proxies suffer a large regret on a significant fraction of downstream tasks. (iv) Finally, we develop a hybrid approach which generalizes across model pools as a practical alternative.

2. Background and related work

We will now introduce the main concepts behind the considered transfer learning approach where the pre-trained model is adapted to the target task by learning a mapping from the intermediate representation to the target labels [21, 29, 31, 32], as illustrated in Figure 1.

(I) Upstream models. The variety of data sources, losses, neural architectures, and other design decisions, leads to a variety of *upstream models*. The user has access to these models, but cannot control any of these dimensions, nor access the upstream training data.

(II) Model search. Given no limits on computation, the problem becomes trivial – exhaustively fine-tune each model and pick the best performing one. In practice, however, one is often faced with stringent requirements on computation, and the aim of the second stage in Figure 1 is therefore to select a relatively small set of models for fine-tuning. Selecting this set of models is the central research question of this paper (cf. Figure 2).

(II A) Task-agnostic search strategies. These approaches rank models before observing the downstream data [14] and consequently select the same model for *every* task. The most popular approach can be summarized as follows (i) Pick the model with the highest test accuracy on ImageNet, otherwise (ii) pick the one trained on the largest dataset. If there is an (approximate) tie, pick the model with most parameters.

(II B.1) Task-aware search strategies. In contrast to the task-agnostic approach, task-aware methods may use the downstream data, thus requiring additional computation. The idea is to extract the learned representations from the pretrained model, train a linear or a kNN classifier on those representations, and select the models that achieve highest accuracy. Note that this requires at least one forward-pass for each instance of the dataset, and for every model, so the computational complexity grows linearly with the model pool size. Nevertheless, this approach is usually several orders of magnitude faster than fine-tuning. Strictly speaking, one can apply an early stopping strategy to the fine-tuning procedure which allows a finer control of the accuracy-time tradeoff. We study this in Section 5.4, whereas in Section 6 we discuss and contrast with other task-aware search strategies.

(II B.2) Meta-learned task-aware search strategies. The idea is to calculate the transfer learning performance for each task in a benchmark set. Given a new dataset, the model search part is performed by finding the best model(s) of the nearest benchmark task, for instance given by learned task embedding on all the benchmark tasks and the downstream dataset. While a single task embedding can be computed efficiently by training a single probe-network, meta-learned approaches require to initially evaluate the performance of all models across each benchmark task. In addition, the list of transfer learning performances needs to be expanded whenever either new models or new benchmark datasets are added. We have simulated this setting by considering the most prominent such approach, namely Task2Vec [1], and provide some preliminary results in Appendix H. In Section 6, we elaborate this approach, along with other related work in this category (e.g. such as semantic-based search strategies [6, 27, 34]), and contrast it to the search strategies we consider in this work.

(III) Downstream training. In this stage, the selected model is adapted to the downstream task (cf. Figure 1). The predominant approach is to fully or partially apply the pre-trained neural network as a feature extractor. The head (e.g. last linear layer) of the pre-trained model is replaced with a new one, and the whole model is trained on the target data. This process is commonly referred to as *fine-tuning* and it often outperforms other methods [5, 14, 20, 25].

3. Computational budget and regret

The main aim of this work is the study of simple methods to filter and search pre-trained models before stepping into the –more expensive– fine-tuning process. Formally, we define a search method $m(\mathcal{M}, \mathcal{D})$ with budget B as a function which takes a set of models $\mathcal{M}$ and a downstream dataset $\mathcal{D}$ as input, and outputs a set of distinct models $\mathcal{S}_m \subseteq \mathcal{M}$, with $|\mathcal{S}_m| = B$. Those B models are then all fine-tuned to obtain the best test accuracy on the downstream task $\mathcal{D}$.

Budget and regret. Fine-tuning represents the largest computational cost so, accordingly, we define the *number* of models that are fine-tuned as the computational complexity of a given method. Given any fixed budget B, we would like to return a set $\mathcal{S}$ which includes the models resulting in good performance downstream. In particular, we define the notion of *absolute regret* of a search strategy m and a pool of models $\mathcal{M}$ on dataset $\mathcal{D}$ as

$$\underbrace{\max_{m_i \in \mathcal{M}} \mathbf{E}[t(m_i, \mathcal{D})]}_{\text{ORACLE}} - \underbrace{\mathbf{E}\left[\max_{s_i \in \mathcal{S}_m} t(s_i, \mathcal{D})\right]}_{s(m)}, \quad (1)$$

where $t(m, \mathcal{D})$ is the test accuracy achieved when fine-tuning model m on dataset $\mathcal{D}$. The first expectation is taken over the randomness in the $t(\cdot)$ operator, that is, the randomness in fine-tuning and due to a finite sampled test set. In addition to the randomness in $t(\cdot)$, the second expectation also accounts for any potential randomization in the algorithm that computes $\mathcal{S}_m$. We define $s(m)$ as the expected maximal test accuracy achieved by any model in the set $\mathcal{S}_m$, the set of models returned by a fixed strategy m. In our case, kNN is deterministic as all the downstream data
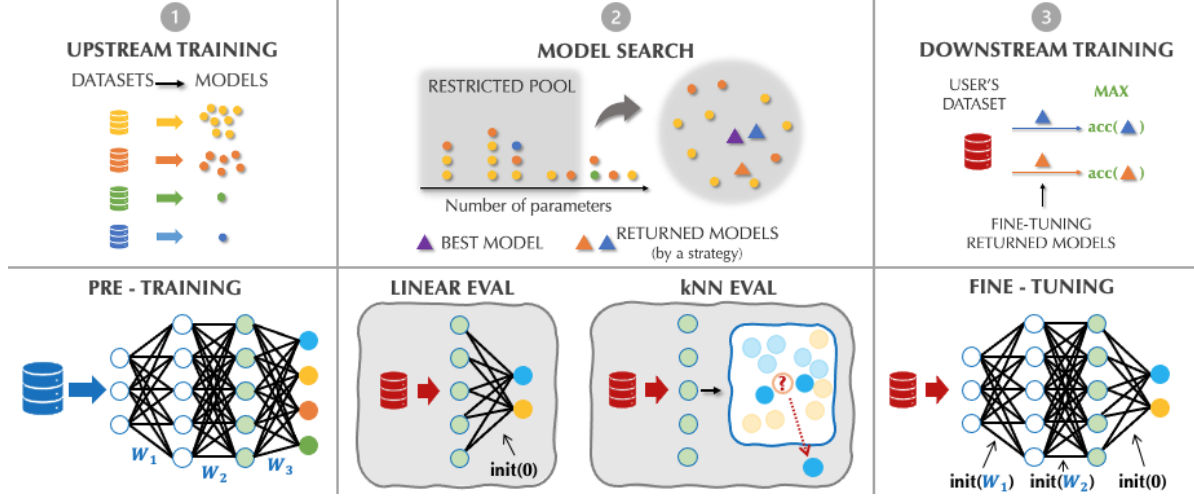
Figure 1. Transfer learning setup: **(1) Upstream models** Pre-training of models from randomly initialized weights on the (large) upstream datasets; **(2) Model search** Either downstream task independent or by running a proxy task, i.e. fixing the weights of all but the last layer and training a *linear classifier* or deploying a *kNN classifier* on the downstream dataset; **(3) Downstream training** Unfreezing all the weights, optimizing all layers (incl. pre-defined ones) on the downstream dataset.
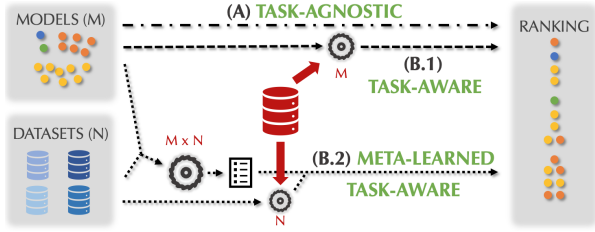

Figure 2. Model search methods: **(A) Task-agnostic** methods do not see the downstream task, producing the same ranking of models for all possible tasks (e.g. using the highest ImageNet accuracy); **(B.1) Task-aware** methods deploy a proxy (e.g. linear evaluation) for each model on user's dataset ; **(B.2) Meta-learned task-aware** methods use a collection of datasets ahead of time for the subsequent model search (e.g. Task2Vec [1], exploring dataset similarities and copying the ranking of the closest dataset).

is used, whereas the linear model depends on the randomness of stochastic gradient descent. To enable comparability between datasets of different difficulty as well as a comparison between two selection strategies m_1, and m_2, we define their *relative delta* as

$$\Delta(m_1, m_2) := \frac{s(m_1) - s(m_2)}{1 - \min(s(m_1), s(m_2))}, \qquad (2)$$

with $s(\cdot) \in [0, 1]$ as defined in Equation 1. Substituting $s(m_1)$ by the ORACLE value, and $s(m_2)$ by $s(m)$ leads to the *relative regret* $r(m)$. We discuss the impact of alternative notions in Section 5.4.

4. Experimental design

Our goal is to assess which model search strategies achieve low regret when presented with a diverse set of models. As discussed, there are three key variables: (i) The

set of downstream tasks, which serve as a proxy for computing the expected regret of any given strategy, (ii) the *model pool*, namely the set we explore to find low-regret models, and (iii) the transfer-learning algorithms.

4.1. Datasets and models

Datasets. We use VTAB-1K, a few-shot learning benchmark composed of 19 tasks partitioned into 3 groups – •*natural*, •*specialized*, and •*structured* [35]. The *natural* image tasks include images of the natural world captured through standard cameras, representing generic objects, fine-grained classes, or abstract concepts. *Specialized* tasks contain images captured using specialist equipment, such as medical images or remote sensing. The *structured* tasks are often derive from artificial environments that target understanding of specific changes between images, such as predicting the distance to an object in a 3D scene (e.g. DeepMind Lab), counting objects (e.g. CLEVR), or detecting orientation (e.g. dSprites for disentangled representations). Each task has 800 training examples, 200 validation examples, and the full test set. This allows us to evaluate model search strategies on a variety of tasks and in a setting where transfer learning offers clear benefits with respect to training from scratch [35].

Models. The motivation behind the model pools is to simulate several use-cases that are ubiquitous in practice. We collect 46 classification models (cf. Appendix A):

- 15 models trained on the ILSVRC 2012 (ImageNet) classification task [24], including Inception V1-V3 models [28], ResNet V1 and V2 (depth 50, 101, and 152) [10], MobileNet V1 and V2 [11], NasNet [36] and PNasNet [15] networks.
- 16 ResNet-50-V2 models trained on (subsets of)

JFT [22]. These models are trained on different subsets of a larger dataset and perform significantly better on a small subset of downstream tasks we consider (i.e. they can be considered as *experts*).

- 15 models from the VTAB benchmark[2], with a diverse coverage of losses (e.g. generative, self-supervised, self-supervised combined with supervised, etc.) and architectures. In all cases the upstream dataset was ImageNet, but the evaluation was performed across the VTAB benchmark which does not include ImageNet.

4.2. Model pools

(A) Identifying good resource-constrained models (RESNET-50, DIM2048). Here we consider two cases: (i) RESNET-50: All models with the number of parameters smaller or equal to ResNet50-V2. While the number of parameters is clearly not the ideal predictor, this set roughly captures the models with limited memory footprint and inference time typically used in practice. Most notably, this pool excludes the NasNet and PNasNet architectures, and includes the *expert* models. (ii) DIM2048: The transfer strategies discussed in Section 2 might be sensitive to the size of the representation. In addition, restricting the representation size is a common constraint in practical settings. Hence, we consider a model pool where the representation dimension is limited to a maximum of 2048.

(B) Identifying expert models in presence of non-experts (EXPERT). We consider a pool of 16 ResNet-50-V2 models from [22]. These models which we considered as *experts* are trained on different subsets of a larger dataset. As the number of available models and the upstream training regimes increase, the number of such experts is likely to increase. As such, this presents a realistic scenario in which an expert for the target task may be present, but it is hard to identify it due to the presence of other models, some of which might perform well *on average*.

(C) Do better task-agnostic models transfer better (IMNETACCURACIES)? This pool offers the ability to choose an upstream representation-learning technique that is best suited for a specific downstream task. This pool is mainly used to validate the idea that (a) ImageNet models transfer well across different tasks [9,12] and that (b) better ImageNet models transfer better [14].

(D) All models (ALL). Finally, we consider the hardest setting, namely when the model pool contains all 46 models and no conceptual nor computational restrictions are in place. We note that: EXPERT ⊂ RESNET-50 ⊂ DIM2048 ⊂ ALL and IMNETACCURACIES ⊂ ALL.

4.3. Evaluation procedures

Fine tuning. To assign a downstream test accuracy to each pair of model and task, we use the median test performance of 5 models obtained as follows: (i) Add a lin-

ear layer followed by a softmax layer and train a model on all examples of the training set. (ii) Fine-tune the obtained model twice, considering two learning rates, and 2500 steps of SGD and a batch size of 512 [35]. (iii) Return the model with the highest validation accuracy. Note that in this case, the entire model, and not only the linear layer, is retrained. As a result, there are 10 runs for each model and we obtain 8740 trained models ($46 \times 19 \times 5 \times 2$).

Linear evaluation. We train a logistic regression classifier added to the model representations (fixed) using SGD. We consider two learning rates (0.1 and 0.01) for 2500 steps and select the model with the best validation accuracy. For robustness we run this procedure 5 times and take the median validation accuracy out of those resulting values. As a result, we obtain again 8740 models.

k**NN evaluation.** We compute the validation accuracy by assigning to each of the 200 validation samples the label of the nearest training example (i.e. $k = 1$) using standard Euclidean distance.

5. Key experimental results

In this section we challenge common assumptions and highlight the most important findings of this study, whilst the extended analysis containing all the plots and tables can be found in the supplementary material. We remark that in the main body we only consider three main pools of models – ALL, RESNET-50 and EXPERT, as we see them as the most representative ones. Since DIM2048 behaves very similarly to RESNET-50, whereas IMNETACCURACIES is used only to confirm the findings of [14], the results of ablation studies involving these two pools can be found in Appendix D. Finally, in this section we mainly investigate linear evaluation as the task-aware choice; all the corresponding plots for kNN can be found in Appendix E.

5.1. High regret of task-agnostic strategies

Figure 3 shows the results for task-agnostic methods with budget $B = 1$ and $B = 2$ on the ALL, RESNET-50, and EXPERT pools. We observe a significant regret, particularly for RESNET-50 and EXPERT pools (30% of the datasets have a relative regret larger than 25% on those two pools). This highlights the fact that task-agnostic methods are not able to pick expert models, in particular on natural and structured datasets. As more experts become available, this gap is likely to grow, making it clear that task-agnostic strategies are inadequate on its own.

5.2. Are task-aware always a good predictor?

Intuitively, having access to the downstream dataset should be beneficial. We evaluate both the linear and the kNN predictor as detailed in Section 4. Figure 4 provides our overall results for the linear model, whereas analogous results for kNN are presented in Appendix E. The method struggles on some structured datasets (in particular on DSPR-LOC).
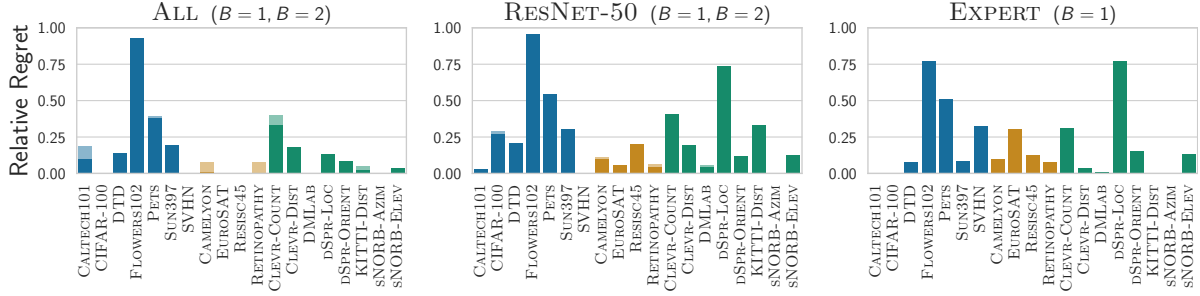
---

[2]<https://tfhub.dev/vtab>

Figure 3. **Task-agnostic strategies.** Relative regret ($r(m)$, cf. Section 3) with $B = 1$ (transparent) and $B = 2$ (solid) on the ALL, RESNET-50 and EXPERT pools, bearing in mind that there is only one task-agnostic model in EXPERT. By definition, task-agnostic strategies exclude experts yielding high regret on the RESNET-50 and EXPERT pools, particularly on natural or structured datasets.
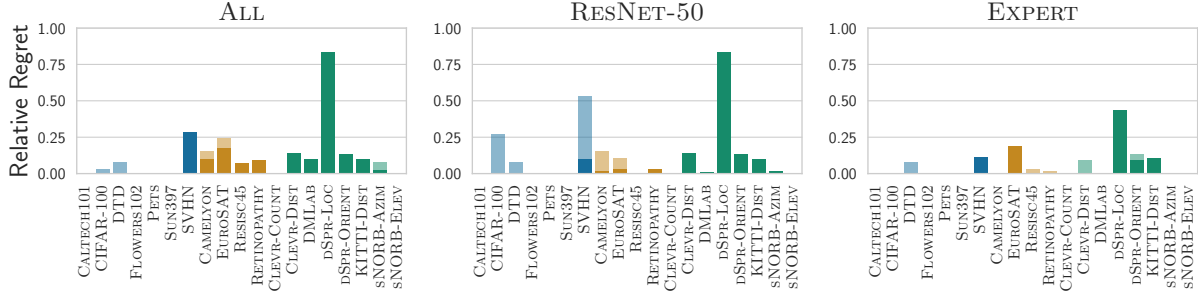

Figure 4. **Task-aware strategies (linear).** Relative regret for $B = 1$ (transparent) and $B = 2$ (solid) on the ALL, RESNET-50, and EXPERT pools. Compared to task-agnostic strategies, we observe improvement on natural datasets (except SVHN) and on restricted pools (except DSPR-LOC), due to its ability to properly choose experts.

Compared to the task-agnostic strategy, as presented in Figure 5 for $B = 1$, we observe significant improvements on restricted model pools. The EXPERT pool benefits the most: linear evaluation outperforms task-agnostic methods on almost every dataset (task-aware is only outperformed on three datasets by more than 1%, and by 10% in the worst case on the KITTI-DIST dataset). On the other hand, task-agnostic and task-aware strategies seem to outperform each other on a similar number of datasets and by a comparable magnitude in the ALL pool. This suggests that no single strategy uniformly dominates all the others across pools.

In order to understand this further, we perform an ablation study where we plot the linear and kNN regret on the IMNETACCURACIES pool in Appendix D. In Figures 15 and 16 we observe that task-aware search methods perform rather poorly when having access *only* to different architectures trained on the same upstream data. The IMNETACCURACIES models are included in the ALL pool, and in some datasets some of those models are the best-performing ones.

Performance of the kNN predictor is on par on half of the datasets across the pools, and slightly worse than linear evaluation on the other half. We present these findings in Figure 20 in Appendix E.

5.3. Hybrid approach yields the best of both worlds

A hybrid approach that selects both the top-1 task-agnostic model and the top-$(B-1)$ task-aware models leads to strong overall results. Figure 6 shows how the hybrid approach with linear evaluation as the task-aware method significantly outperforms its linear counterpart with $B = 2$. This is most noticeable in the ALL pool where the task-agnostic model provides a large boost on some datasets.

As we saw in Figure 5, when looking at the ALL pool, the task-agnostic candidate tends to beat the linear one on datasets such as DSPR-LOC, SVHN or EUROSAT. Similarly, the linear candidate model clearly outperforms its task-agnostic counterpart on many natural datasets such as FLOWERS or PETS. A comparison of Figures 5 and 6 reflects how the dominance of linear-only strategy vanishes on most datasets when confronted with the hybrid approach. For the RESNET-50 and EXPERT pools, as expected, the hybrid algorithm preserves the good picks of the linear proxy. That said, we observe an increase of 36% on DSPR-LOC in the RESNET-50, and 11% on KITTI-DIST. Both are structured datasets on which the linear proxy task performs poorly, as shown in Figure 4.

The hybrid strategy requires to fine-tune at least two models. Given that it performs well across all model pools and datasets, this is a reasonable price to pay in practice, and we suggest its use as the off-the-shelf approach. Figures 21 and 22 in Appendix E depict the results for kNN. In EXPERT models, the second kNN pick tends to beat the task-agnostic one – hurting the kNN hybrid outcomes. Overall, the hybrid linear approach consistently outperforms the one based on kNN.
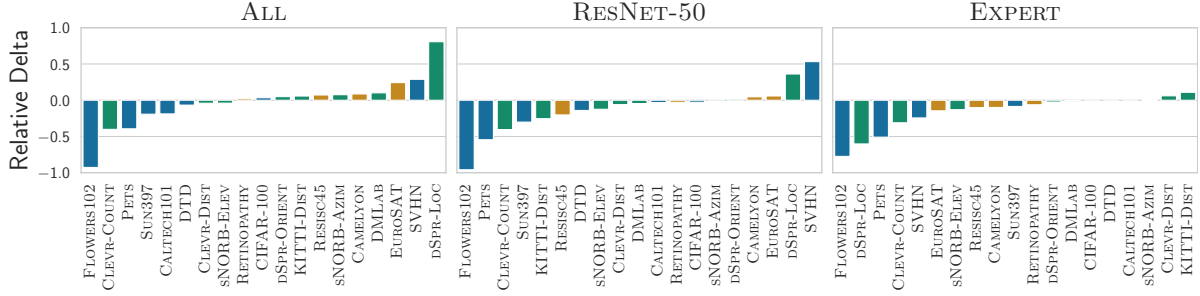
Figure 5. **Task-agnostic** (positive if better) **vs Task-aware (linear)** (negative if better) for $B = 1$. On the ALL pool, the methods perform in a similar fashion, with respect to the number of datasets and the amount in which one outperforms the other. When one restricts the pool to RESNET-50 or EXPERT task-aware methods outperform the task-agnostic method on most datasets. The relative delta is defined in Equation 2 in Section 3.
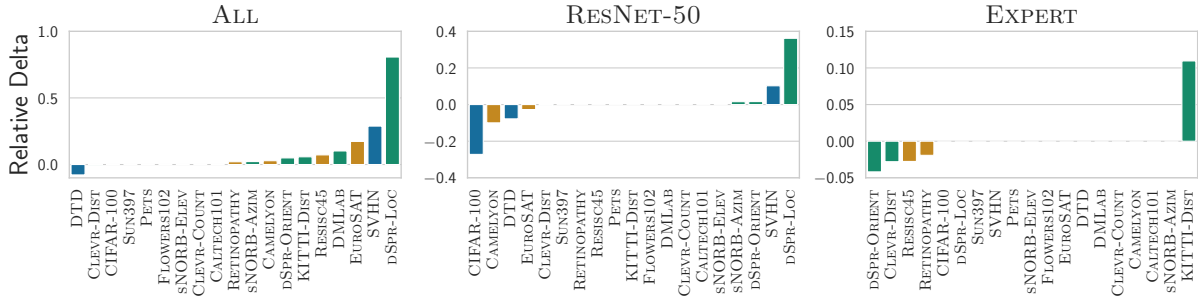

Figure 6. **Hybrid linear** (positive if better) **vs Linear evaluation** (negative if better) for $B = 2$. We observe that hybrid linear significantly outperforms linear with the same budget on the ALL pool. Even though for RESNET-50 and EXPERT pools there are datasets on which linear performs better than hybrid, the amounts in which it does are usually small. We note that most significant gains of hybrid come on certain structured datasets, the hardest task for every strategy.

5.4. Further ablation studies

How does the computational budget impact the findings? We have seen that for a limited budget of $B = 2$ the proposed hybrid method outperforms the other strategies. A natural question that follows is: how do these methods perform as a function of the computational budget B? In particular, for each budget B, we compute how frequently does a strategy pick the best model. The results are shown in Figure 7. We observe that the hybrid linear strategy outperforms all individual strategies on the ALL pool. Furthermore, it also outperforms a strong impractical task-agnostic oracle which is allowed to rank the models by the average fine-tune accuracy over all datasets. Our hybrid strategy achieves an on par performance with the linear approach on pools on which linear performs well. When task-aware strategies perform badly (e.g. pools without expert models), hybrid linear is significantly stronger (as seen in Figure 17 in Appendix D). These empirical results demonstrate that the hybrid strategy is a simple yet effective practical choice for a model search strategy.

Alternative evaluation scores. Both [14] and [16] compute the correlation between the ImageNet test accuracy and the average fine-tune accuracy across datasets. Although this provides a good task-agnostic evaluation method for the *average performance* (cf. Figure 1, right, in

[14]), it can be significantly impacted by outliers that have poor correlations on a specific dataset (cf. Figure 2, middle row, in [14]). In Appendix B we highlight another limitation of using correlation scores in the context of model-search strategies across heterogeneous pools. Nevertheless, we empirically validate that ranking the models based on their ImageNet test accuracy on the IMNETACCURACIES pool transfers well to our evaluation setting (cf. Figure 14 in the Appendix D). Furthermore, we show that reporting the differences of logit-transformed accuracies (log-odds) leads to similar conclusions as ours (cf. Appendix C). We opt for the relative regret $r(m)$, defined in Section 3, as it is more intuitive and contained in $[-1, 1]$.

Impact of the kNN hyperparameters. The kNN classifier suffers from the curse of dimensionality [26], which is why we study the impact of the dimension (i.e. the representation size) on the kNN evaluation. We fix a dataset, and plot a model's kNN score versus its representation dimension. In order to have a single point per dimension and avoid an over-representation of the expert models that are all of the same architecture, we choose the model with the best kNN accuracy. By calculating the Pearson correlation coefficient between the dimension and the respective kNN scores, we observe a moderate anti-correlation ($R < -0.5$) for 3 datasets, a moderate correlation ($R > 0.5$) for 3 other
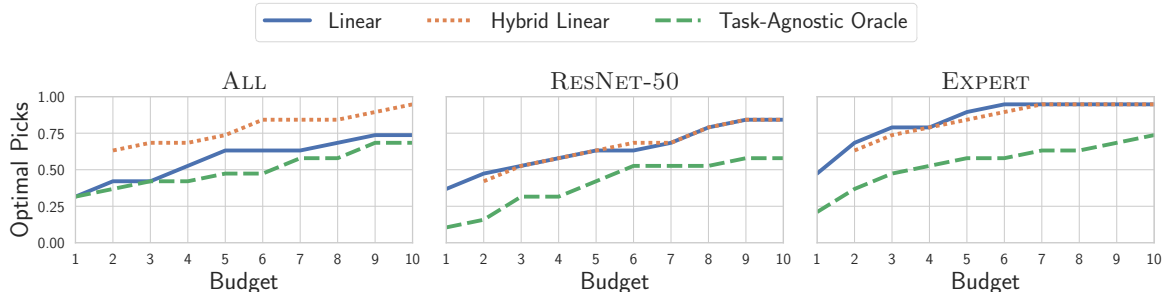
Figure 7. **Optimal picks as a function of the computational budget.** The number of picked models (relative) with zero regret across three representative pools. We note that hybrid linear outperforms all other methods on ALL, whilst being comparable with the linear strategy on restricted pools where linear alone already performs well. Here, the task-agnostic oracle refers to a method which ranks models based on their average accuracy across all datasets (more details Section 5.4).

datasets, and either small or no correlation for the remaining 13 datasets. Based on this empirical evidence we conclude that there is no significant correlation between the kNN classifier accuracy and the dimension. We provide more details in Figure 23 of Appendix F. Regarding k, our preliminary experiments with $k = 3$ offered no significant advantages over $k = 1$.

Early stopping and fine-tuning. *Early stopping* approach, namely fine-tuning the entire network only for a small number of iterations, allows one to explore the accuracy vs. time tradeoff. However, it is understood that results heavily depend on the neural architectures and hyperparameters used during pre-training (e.g. models trained with Batch normalization necessitate a different strategy). In our experiments, summarized in Appendix I, we do not exhibit any benefits which generalize beyond certain specific settings that would favor this strategy over the simple linear proxy (cf. Figures 28).

6. Other related work

Given access to the meta-data such as the source datasets, one could compare the upstream and downstream datasets [3]; blend the source and target data by reweighting the upstream data to reflect its similarity to the downstream task [18]; or construct a joint dataset by identifying subsets of the upstream data that are well-aligned with the downstream data [8]. These approaches are restricted to one model per upstream dataset, while also being less practical as they necessitate training a new model as well as access to upstream datasets, which might be unavailable due to proprietary or privacy concerns.

Additionally, best-arm identification bandits algorithms suggest the successive elimination of sub-optimal choices [7] derived by fine-tuning models for shorter time. This combines model selection and downstream training, noting that it suffers from the same limitations as the partial fine-tuning proxy approach described Section 5.4.

Other task-aware strategies. Alternative approaches such as H-Score [2], LEEP [19] or NCA [30] replace the

computation of the linear classifier's weights on top of the frozen weights by using a cheaper estimator for the classifier's accuracy. These estimators are derived using a pseudo label distribution by analyzing the classification output of the pre-trained models. All three approaches have a clear limitation in supporting only those cases in which a classification head for the original task is provided together with each pre-trained model. Furthermore, the theoretical guarantees are given with respect to linear classification models trained on top of the frozen representations only. As observed with LEEP, albeit on natural datasets only, fine-tuning achieves significantly better test accuracies than a linear classifier, which further invites for better understanding of using linear accuracy as a proxy for fine-tuning, noting that linear correlations are not necessarily transitive. Furthermore, training a single linear layer, or calculating the kNN accuracy on top of transformed representations, is typically much less computational demanding than running the inference itself, and both steps can easily be overlapped. Recently, [16] introduced the Mean Silhouette Coefficient (MSC) to forecast a model's performance after fine-tuning on time series data. We omit this approach due to its provable relation to a linear classifier proxy, whilst being similar to kNN with respect to capturing non-linear relationships. Finally, [22] utilize kNN as a cheap proxy task for searching models in a set of experts with the same architecture without quantifying and evaluating the regret. LFC/LGC [4] are comparable to the early stopping strategy, which instead of using the gradient at the initialization point to compare models, follows this gradient for a small number of iterations. We note that this related work evaluates its proposed strategy in a fairly homogeneous model pool, a setting in which many task-aware methods and the corresponding early-stopping strategies have less room to fail.

Meta-learned strategies. Meta-learned search strategies aim at reducing the linear complexity of having to run inference and training a subsequent model with the downstream dataset on all pre-trained models for performing a model search. Such approaches typically contain a

meta-learning part that precedes the actual search component by making use of a set of benchmark datasets. These datasets do not have to be related to the datasets used to train models upstream. In fact, in our scenario, the upstream datasets are unknown or inaccessible to the model search method, thereby rendering the benchmark datasets independent from the pre-trained models. The computational complexity of performing the search is moved to the meta-learning part by calculating the task embedding and fine-tuning all pre-trained models on a set of benchmark datasets, as described in Section 2 for Task2Vec [1]. This computationally intensive task could be re-used by subsequent model search queries with different downstream datasets, but would typically need to be maintained by the provider of pre-trained model repositories. We simulate the performance of Task2Vec in our experimental setting in Appendix H. We observe that meta-learned task-aware strategies can be improved by task-agnostic search methods (cf. Figure 25) for the budget of $B = 1$, but not necessarily for a larger budget. Investigating the impact of benchmark datasets and larger fine-tune budget for these methods is beyond the scope of this work and represents an interesting line of future work.

Another line of related work is Taskonomy [34] which finds the best source dataset(s), or nearest benchmark task in the meta-learned context, by exhaustively exploring the space of all possibilities. While it achieves promising results, it is not directly comparable to our approach: (i) the input domain and data is assumed to remain constant and tasks are only different in their labels, (ii) the chosen architecture keeps the weights of the encoder frozen, and the decoder trained on top usually consists of multiple fully connected layers, opposed to our fine-tuning regime with a single linear layer on top. Improvements by [6] and [27] make Taskonomy faster, but they still do not distinguish multiple models trained on the same dataset, nor bypass the constraints described previously. Finally, for generalization of this approach beyond classification tasks, Mensink et al. [17] extensively studied transfer behaviour across different types of tasks. They suggest that a task-agnostic strategy of selecting the model fine-tune on the largest possible dataset will perform well, whereas if a task-aware strategy is able to find a model trained on a most similar source, if it exists, it should outperform the task-agnostic strategy.

7. Conclusions, limitations, and future work

Transfer learning offers a data-efficient solution to train models for a range of downstream tasks. As we witness an increasing number of models becoming available in repositories such as TensorFlow Hub, finding the right pre-trained models for target tasks is getting harder. Fine-tuning all of them is not an option. In practice, the computational budget is limited and efficient model search strategies become paramount. We motivate and formalize the problem of efficient model search through a notion of *regret*, and argue that regret is better suited to evaluate search algorithms than correlation-based metrics. Empirical evaluation results for the predominant strategies, namely *task-agnostic* and *task-aware* search strategies, are presented across several scenarios, showing that both can sometimes yield high regret. In other words, for any individual method we study, there exists a pool of models on which the method fails. Finally, we propose a simple and computationally efficient hybrid search strategy which consistently outperforms the existing approaches over 19 diverse vision tasks and across all the defined model pools.

Limitations and future work. In order to further stress-test the generalization of analyzed strategies, the number of relevant model pools could be increased by incorporating more diverse upstream tasks, in particular neural architectures, losses, and datasets. This would potentially yield more expert models, making the task both more challenging and more important, further highlighting the advantages of an effective search strategy. Secondly, we observe that task-aware methods consistently perform poorly in specific cases, such as when we consider diverse architectures trained only on ImageNet. There is no obvious reason for such failures. Similarly, there seems to be a clear pattern where task-aware methods perform significantly worse on structured datasets than on natural ones. We hypothesize that this is due to the lack of adequate expert models for these domains. However, an in-depth analysis of these specific cases might be beneficial and insightful.

Social impact. We examine and propose a mixed model search strategy focusing solely on maximizing downstream accuracy. Applying such a search strategy blindly might enable hurtful biases present in pre-trained models, or in the datasets used to train the models upstream, to propagate into the models trained downstream. Quantifying the impact of transfer learning and different search strategies beyond accuracy is left as future work, whereby we encourage users of online pre-trained models to check for biases and fairness issues after fine-tuning on their custom datasets.

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. *IEEE International Conference on Computer Vision*, pages 6430–6439, 2019. 1, 2, 3, 8, 23

[2] 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. *IEEE International Conference on Image Processing*, pages 2309–2313, 2019. 7

[3] Bishwaranjan Bhattacharjee, John R Kender, Matthew Hill, Parijat Dube, Siyu Huo, Michael R Glass, Brian Belgodere, Sharath Pankanti, Noel Codella, and Patrick Watson. P2l: Predicting transfer learning for images and semantic relations. *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 760–761, 2020. 7

[4] 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. 7

[5] 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. *International Conference on Machine Learning*, 2014. 2

[6] Kshitij Dwivedi, Jiahui Huang, Radoslaw Martin Cichy, and Gemma Roig. Duality diagram similarity: a generic framework for initialization selection in task transfer learning. *European Conference on Computer Vision*, 2020. 2, 8

[7] Eyal Even-Dar, Shie Mannor, and Yishay Mansour. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *Journal of Machine Learning Research*, 2006. 7

[8] Weifeng Ge and Yizhou Yu. Borrowing treasures from the wealthy: Deep transfer learning through selective joint fine-tuning. *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 7

[9] Kaiming He, Ross Girshick, and Piotr Dollár. Rethinking imagenet pre-training. *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 4

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 3

[11] 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. 3

[12] Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. What makes ImageNet good for transfer learning? *arXiv preprint arXiv:1608.08614*, 2016. 1, 4

[13] Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. Revisiting self-supervised visual representation learning. *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 1

[14] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better Imagenet models transfer better? *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 1, 2, 4, 6, 13, 14, 16

[15] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. *European Conference on Computer Vision*, 2018. 3

[16] Amiel Meiseles and Lior Rokach. Source model selection for deep learning in the time series domain. *IEEE Access*, 2020. 1, 6, 7, 13

[17] Thomas Mensink, Jasper Uijlings, Alina Kuznetsova, Michael Gygli, and Vittorio Ferrari. Factors of influence for transfer learning across diverse appearance domains and task types. *arXiv preprint arXiv:2103.13318*, 2021. 8

[18] Jiquan Ngiam, Daiyi Peng, Vijay Vasudevan, Simon Kornblith, Quoc V Le, and Ruoming Pang. Domain adaptive transfer learning with specialist models. *arXiv preprint arXiv:1811.07056*, 2018. 1, 7

[19] Cuong V Nguyen, Tal Hassner, Cedric Archambeau, and Matthias Seeger. Leep: A new measure to evaluate transferability of learned representations. *International Conference on Machine Learning*, 2020. 7

[20] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. *IEEE Conference on Computer Vision and Pattern Recognition*, 2014. 1, 2

[21] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 2009. 2

[22] Joan Puigcerver, Carlos Riquelme, Basil Mustafa, Cedric Renggli, André Susano Pinto, Sylvain Gelly, Daniel Keysers, and Neil Houlsby. Scalable transfer learning with expert models. *International Conference on Learning Representations*, 2021. 1, 4, 7, 11

[23] Maithra Raghu, Chiyuan Zhang, Jon Kleinberg, and Samy Bengio. Transfusion: Understanding transfer learning for medical imaging. *Advances in Neural Information Processing Systems*, 2019. 1

[24] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 2015. 3

[25] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: An astounding baseline for recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, 2014. 2

[26] Robert R Snapp, Demetri Psaltis, and Santosh S Venkatesh. Asymptotic slowing down of the nearest-neighbor classifier. *Advances in Neural Information Processing Systems*, 1991. 6

[27] Jie Song, Yixin Chen, Jingwen Ye, Xinchao Wang, Chengchao Shen, Feng Mao, and Mingli Song. Depara:

Deep attribution graph for deep knowledge transferability. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3922–3930, 2020. 2, 8

[28] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the Inception architecture for computer vision. *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 3

[29] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. *International Conference on Artificial Neural Networks*, 2018. 2

[30] Anh T Tran, Cuong V Nguyen, and Tal Hassner. Transferability and hardness of supervised classification tasks. *IEEE International Conference on Computer Vision*, pages 1395–1405, 2019. 7

[31] Zirui Wang. Theoretical guarantees of transfer learning. *arXiv preprint arXiv:1810.05986*, 2018. 2

[32] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 2016. 2

[33] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in Neural Information Processing Systems*, 2014. 1

[34] Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 2, 8

[35] Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, et al. The Visual Task Adaptation Benchmark. *arXiv preprint arXiv:1910.04867*, 2019. 1, 3, 4

[36] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 3