A. Pre-Trained model details

We provide a detailed list of all the used pre-trained models together with the dimension of their representations, the number of parameters, and the achieved ImageNet test accuracy (for those that the accuracy is known), in the following tables.

Model Name	Dim	# Params	ImageNet Accuracy
inception_v1/feature_vector/4	1024	5'592'624	0.698
inception_v2/feature_vector/4	1024	10'153'336	0.739
inception_v3/feature_vector/4	2048	21'768'352	0.78
inception_resnet_v2/feature_vector/4	1536	54'276'192	0.804
resnet_v1_50/feature_vector/4	2048	23'508'032	0.752
resnet_v1_101/feature_vector/4	2048	42'500'160	0.764
resnet_v1_152/feature_vector/4	2048	58'143'808	0.768
resnet_v2_50/feature_vector/4	2048	23'519'360	0.756
resnet_v2_101/feature_vector/4	2048	42'528'896	0.77
resnet_v2_152/feature_vector/4	2048	58'187'904	0.778
mobilenet_v1_100_224/feature_vector/4	1024	3'206'976	0.709
mobilenet_v2_100_224/feature_vector/4	1280	2'223'872	0.718
nasnet_mobile/feature_vector/4	1056	4'232'978	0.74
nasnet_large/feature_vector/4	4032	84'720'150	0.827
pnasnet_large/feature_vector/4	4320	81'736'668	0.829

Table 1. ImageNet Classification Models – All models are accessible by using the same prefix "https://tfhub.dev/google/imagenet/" in front of the model name.

Model (Subset)	Dim	# Params
Mode of transport	2048	23'807'702
Geographical feature	2048	23'807'702
Structure	2048	23'807'702
Mammal	2048	23'807'702
Plant	2048	23'807'702
Material	2048	23'807'702
Home & garden	2048	23'807'702
Flowering plant	2048	23'807'702
Sports equipment	2048	23'807'702
Dish	2048	23'807'702
Textile	2048	23'807'702
Shoe	2048	23'807'702
Bag	2048	23'807'702
Paper	2048	23'807'702
Snow	2048	23'807'702
Full JFT	2048	23'807'702

Table 2. Expert Models – The model name indicates the subset of JFT on which each model was trained [22].

Model Name	Dim	# Params
sup-100/1	2048	23'500'352
rotation/1	2048	23'500'352
exemplar/1	2048	23'500'352
relative-patch-location/1	2048	23'500'352
jigsaw/1	2048	23'500'352
semi-rotation-10/1	2048	23'500'352
sup-rotation-100/1	2048	23'500'352
semi-exemplar-10/1	2048	23'500'352
sup-exemplar-100/1	2048	23'500'352
cond-biggan/1	1536	86'444'833
uncond-biggan/1	1536	86'444'833
wae-mmd/1	128	23'779'136
wae-gan/1	128	23'779'136
wae-ukl/1	128	23'779'136
vae/1	128	23'779'136

Table 3. VTAB Benchmark Models – All models are accessible by using the model name and the prefix "https://tfhub.dev/vtab/".

B. Limitation of correlation as evaluation score

Previous works suggests to perform a correlation analysis for choosing which model to transfer based on a ranking across the models [14, 16]. We claim that this is not a suitable score in our setting of heterogenous pools, and in this section we explain the arguments in more details. We provide a simple example in which a correlation analysis fails compared to our notion of regret, which we see as an intuitive notion of failure in this setting.

We start by outlining two obvious dependencies between the two variants:

- Having a perfect correlation (equal to 1) results in zero regret,
- Having zero regret does not necessarily imply a perfect correlation.

The first statement follows by definition, whereas the second statement is justified by the following example: suppose that all models perform identically in terms of the fine-tune accuracy. In this case, every attribute (e.g. proxy task value, or ImageNet accuracy) would yield no correlation with respect to the fine-tune accuracy, although there is clearly zero regret for every imaginable strategy.

Implications. If we have a large pool of models with some outlier models that clearly outperform the others, which are of similar fine-tune accuracy, a search strategy should return one of those *better* model, otherwise it will suffer from a large regret. On the other hand, this setting would usually have no rank nor linear correlation following the reasoning from before. If we restrict the same pool to models performing similarly, it will remain uncorrelated, but every search strategy will result in zero regret. The same scenario holds if single outliers are performing worse than all other models. Both cases are seen often in practice, especially for model pools containing experts. We highlight some examples of this phenomena in Figures 8 and 9, together with some that have positive correlation.

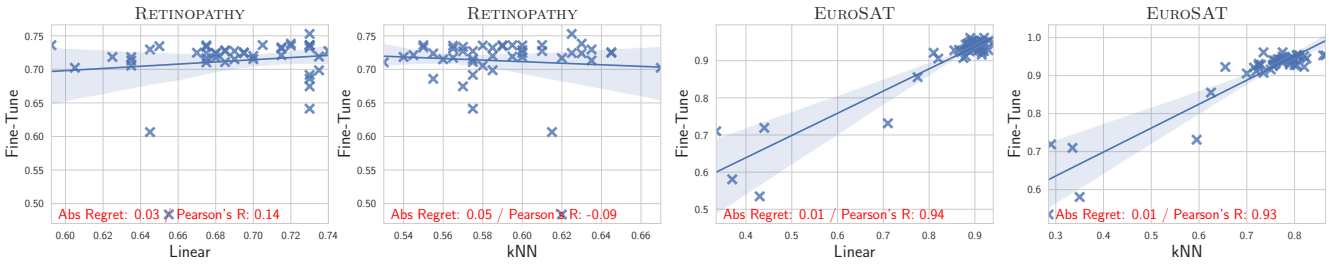

Figure 8. Example correlation values for pool ALL.

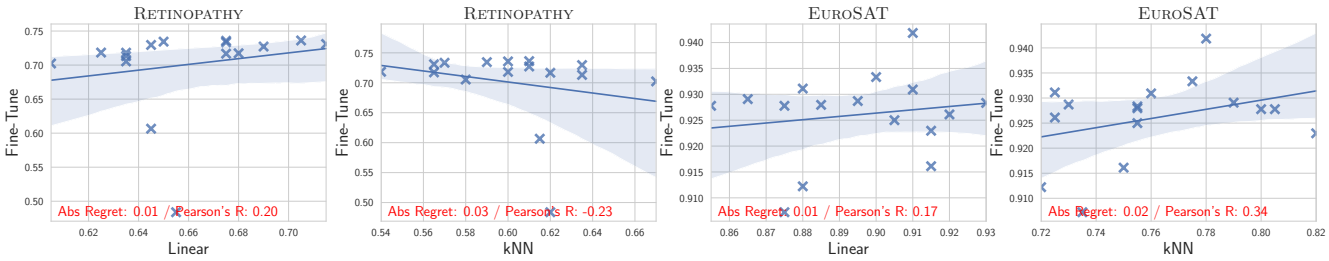

Figure 9. Example correlation values for pool EXPERT.

C. Log-odds for the evaluation score

Following [14], we analyze a notion that differs from our definition of relative regret and delta between two strategies (cf. Equation 2 and Section 3). The idea is to compare two search strategies m_1 and m_2 by calculating the difference between $\text{logit}(s(m_1))$ and $\text{logit}(s(m_2))$ (the expected maximal logit-transformed test accuracy achieved by any model in the sets returned for both search strategies). The logit transform is defined as $\text{logit}(p) = \log(p/(1-p)) = \text{sigmoid}^{-1}(p)$, also known as the log-odds of p. This transformation leads to the next definition of *log-odds delta*:

$$\tilde{\Delta}(m_1, m_2) := \log\left(\frac{s(m_1) - s(m_2)}{1 - \min(s(m_1), s(m_2))}\right). \tag{3}$$

Substituting $s(m_1)$ by the ORACLE value from Equation 1 in Section 3, and $s(m_2)$ by $s(m)$ leads to a new definition of *log-odds regret* $\tilde{r}(m)$.

These definitions are also incorporating the dataset difficulty and yield results very similar to our definition of the relative delta and relative regret in Section 3. We now provide the analogous plots of the ones given in the main body of the paper, with the log-odds regret and log-odds delta instead of the relative regret and relative delta. We highlight the fact that, beside the change of the scale on the y-axis, all the findings given in the main body of the paper hold.

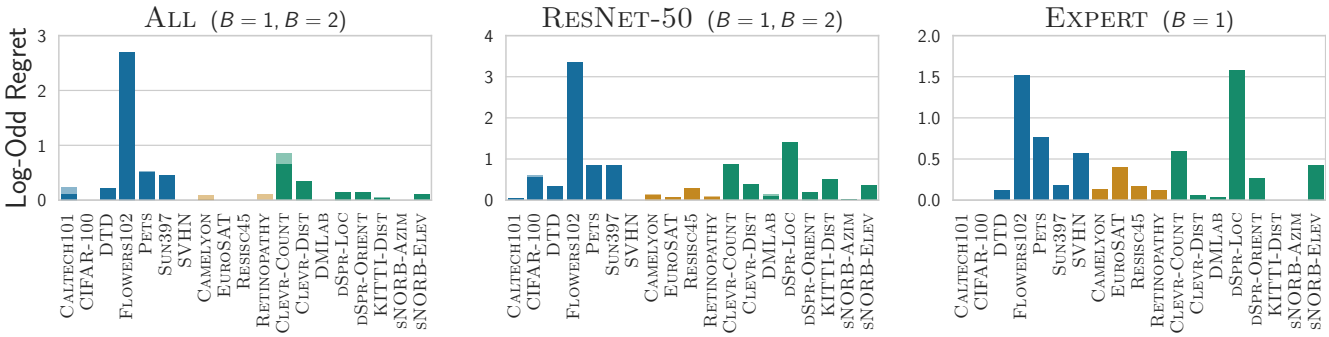

Figure 10. Log-odds regret ($\tilde{r}(m)$) with $B = 1$ (transparent) and $B = 2$ (solid) for the task-agnostic model search strategy.

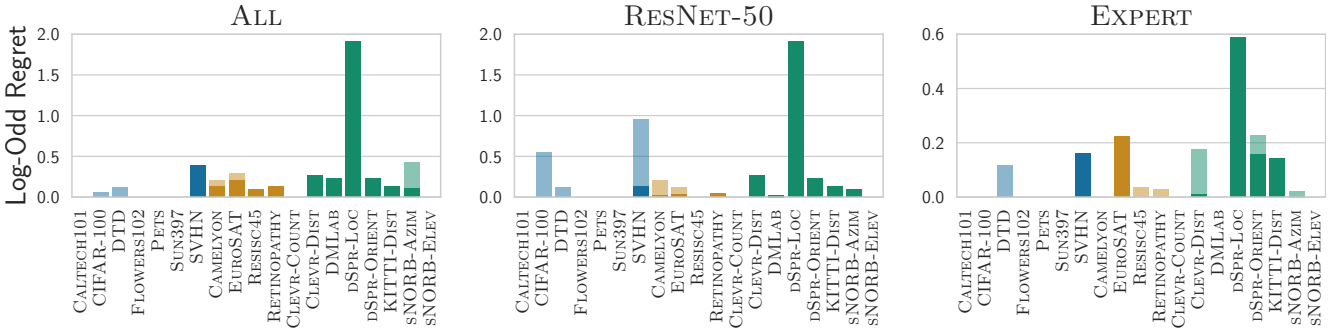

Figure 11. Log-odds regret ($\tilde{r}(m)$) for $B = 1$ (transparent) and $B = 2$ (solid) for the task-aware (linear) model search strategy.

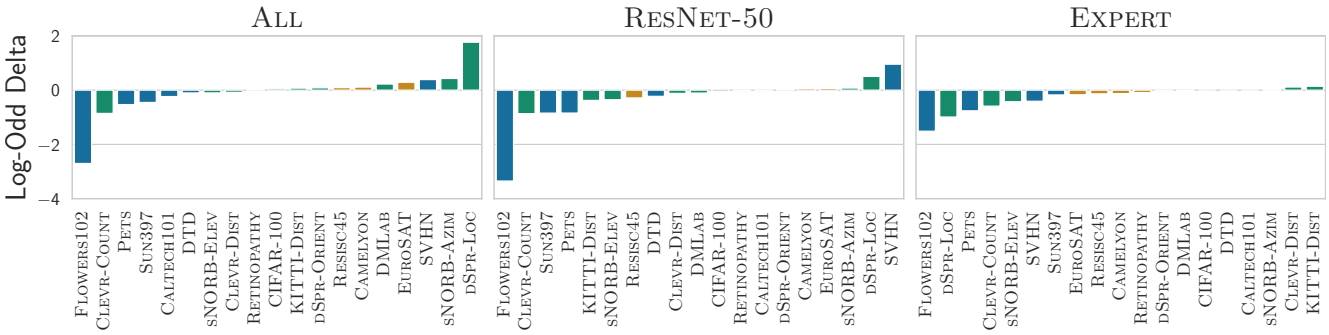

Figure 12. Log-odds delta ($\tilde{\Delta}(m_1, m_2)$) between task-agnostic (positive if better) and task-aware (linear) (negative if better) for $B = 1$.

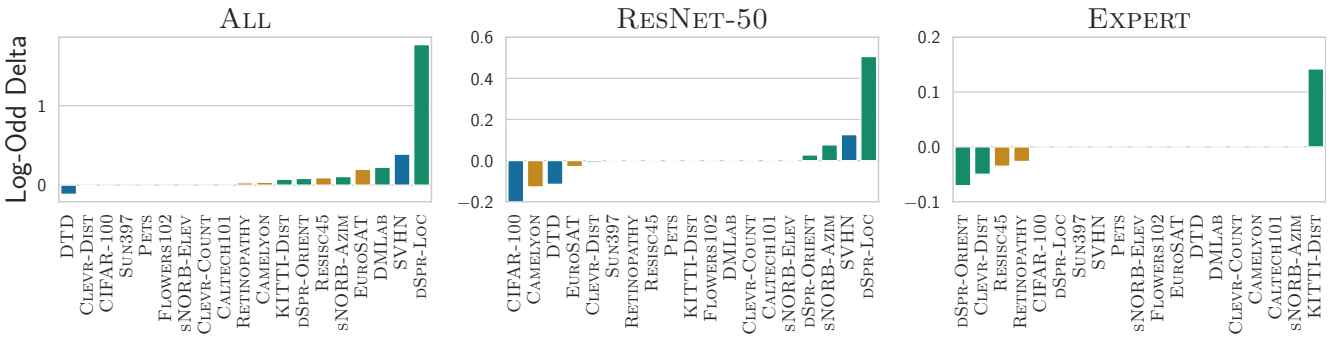

Figure 13. Log-odds delta ($\tilde{\Delta}(m_1, m_2)$) between hybrid linear (positive if better) and linear evaluation (negative if better) for $B = 2$.

D. Analysis for other pools.

In this sections we provide the plots and an analysis for the DIM2048 and IMNETACCURACIES pools, both omitted from the main body of the paper due to space limitations.

We start by emphasizing that the results between the DIM2048 and the RESNET-50 pool, used in the main body of the paper, do not vary significantly. Most notably, the hybrid linear strategy is on par with the task-aware method, whereas the task-agnostic method suffers from high regret due to the lack of ability to pick expert models.

When examining the performance of all strategies on the IMNETACCURACIES pool, presented in Figures 14 (right), 15 (right) and 16 (right), we observe that the task-agnostic strategy is able to pick the optimal model for 12 out of 19 datasets for $B = 1$, and as many as 17 out of 19 for $B = 2$. This clearly confirms the claim made by [14] that *better ImageNet models transfer better*. More surprisingly, we observe that both task-aware strategies (linear and kNN) fail consistently and, hence, result in high regret when being restricted to the IMNETACCURACIES pool only (cf. Figures 15 and 16).

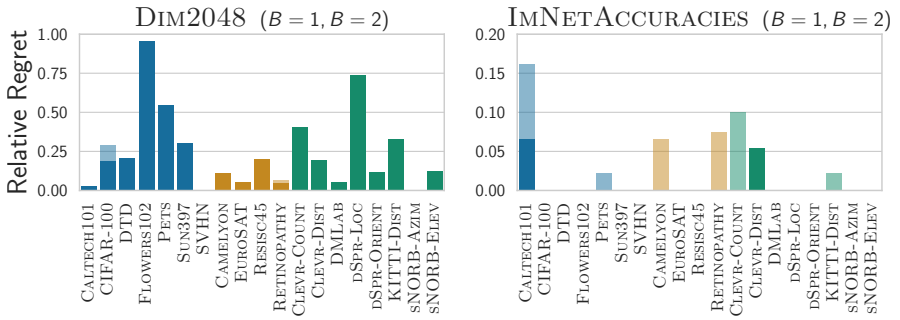

Figure 14. Relative regret for the task-agnostic search strategy with $B = 1$ (transparent) and $B = 2$ (solid) on the pools DIM2048 and IMNETACCURACIES.

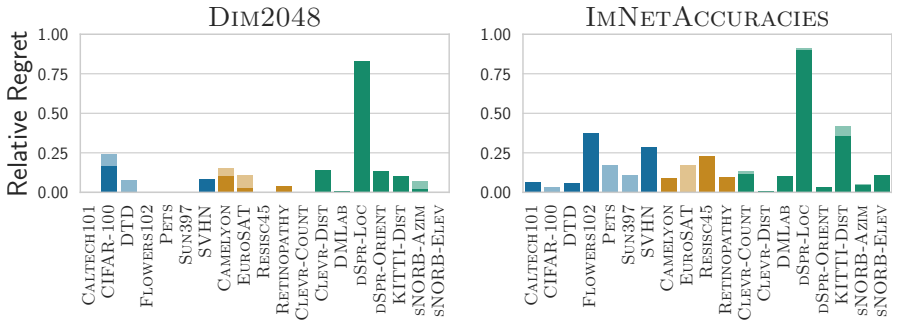

Figure 15. Relative regret for the task-aware (linear) search strategy with $B = 1$ (transparent) and $B = 2$ (solid) on the pools DIM2048 and IMNETACCURACIES.

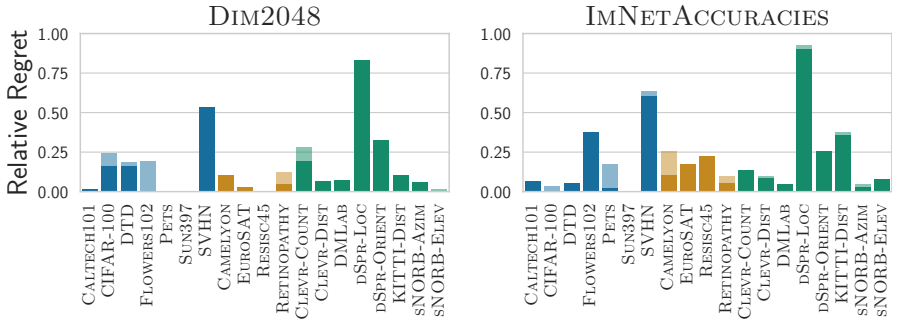

Figure 16. Relative regret for the task-aware (kNN) search strategy with $B = 1$ (transparent) and $B = 2$ (solid) on the pools DIM2048 and IMNETACCURACIES.

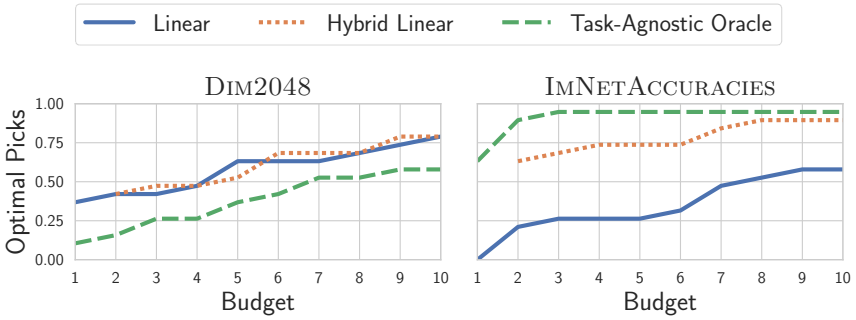

Figure 17. Optimal picks for an increasing budget on DIM2048 and IMNETACCURACIES.

E. kNN as a task-aware proxy

In this section, we analyze the impact of choosing the kNN classifier accuracy as the choice for the proxy task, compared to the linear classifier accuracy described in the main body of the paper. In practice, kNN might be favorable to a user since calculating the kNN classifier accuracy with respect to a relatively small test set can be orders of magnitude faster compared to training a linear classifier, which might be sensitive to the choice of optimal hyper-parameters. A theoretical and empirical analysis of the computing performance of these two proxies is out of the scope of this work, as we are mainly interested in the comparison in terms of the model-search capability of either of these. In general, the major claims on the performance of linear as a task-aware strategy also apply to kNN. The latter also mainly fails on structured datasets across all pools, as visible in Figure 18. Similarly to the linear task, kNN is on par with the task-agnostic strategy on the ALL pool, but clearly outperforms it on the RESNET-50 and EXPERT pools, as visible in Figure 19. By further comparing kNN to the linear proxy task, we realize that kNN performs worse than linear on half of the datasets across the three different dataset groups, whilst being on par with it on the other half (cf. Figure 20). Finally, by choosing kNN as the task-aware part for the hybrid strategy and comparing it to the task-aware (kNN) strategy with a budget of $B = 2$ in Figure 21, we see an increase of performance on the ALL pool, no clear winner on the restricted RESNET-50 pool, but higher regret on the EXPERT pool. Unsurprisingly, this version of hybrid strategy also performs slightly worse compared than the one with a linear proxy across all the pools (cf. Figure 22).

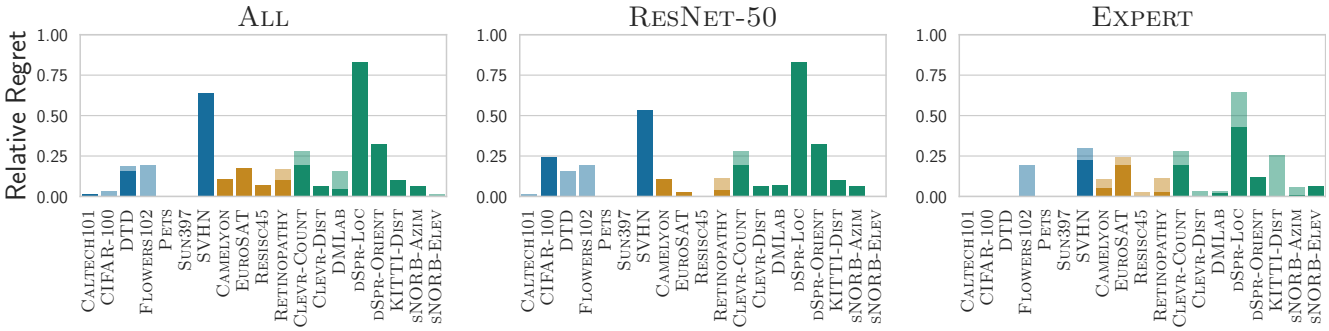

Figure 18. Relative regret for the kNN search strategy with $B = 1$ (transparent) and $B = 2$ (solid).

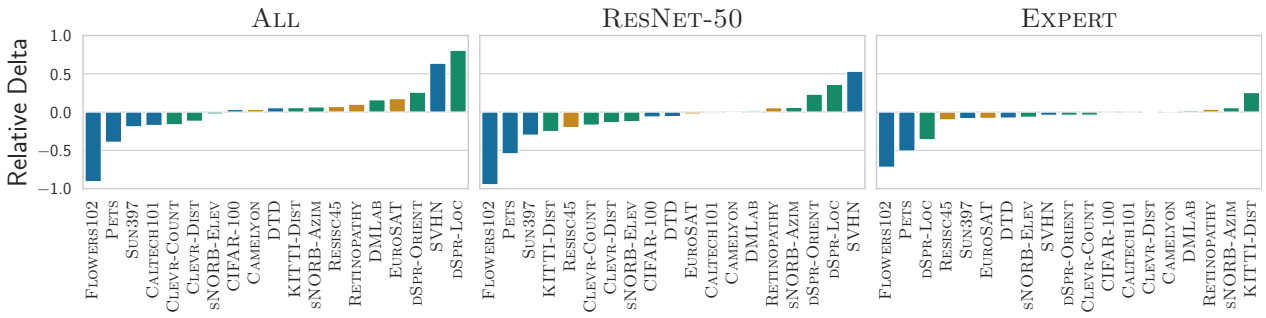

Figure 19. Relative delta between the task-agnostic (positive if better) and the kNN task-aware search strategy (negative if better) for $B = 1$.

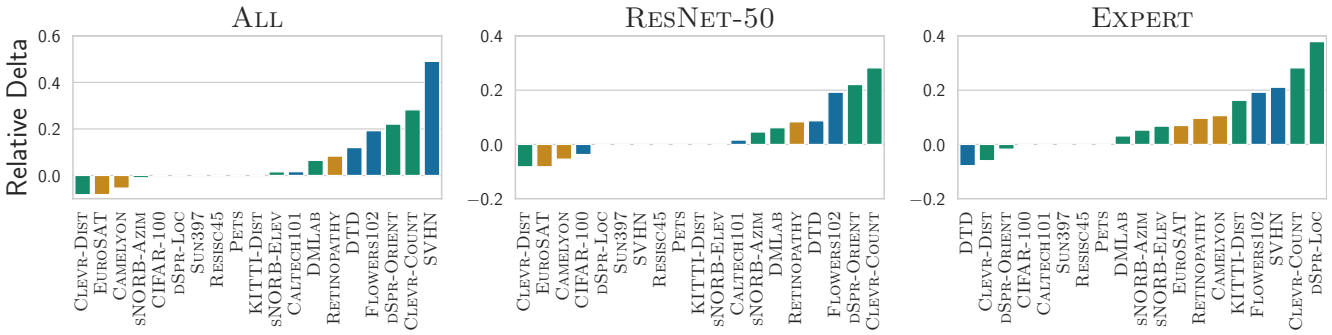

Figure 20. Relative delta between the linear (positive if better) and the kNN task-aware search strategy (negative if better) for $B = 1$.

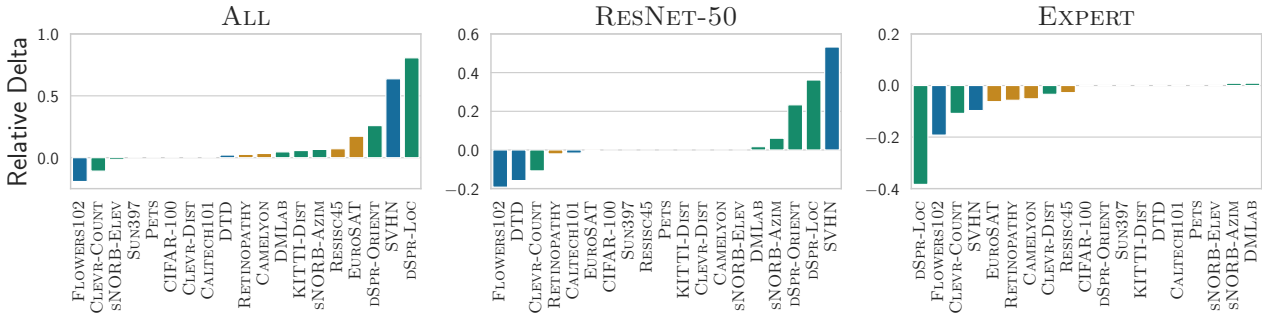

Figure 21. Relative delta between the hybrid kNN (positive if better) and kNN search strategy (negative if better) for $B = 2$.

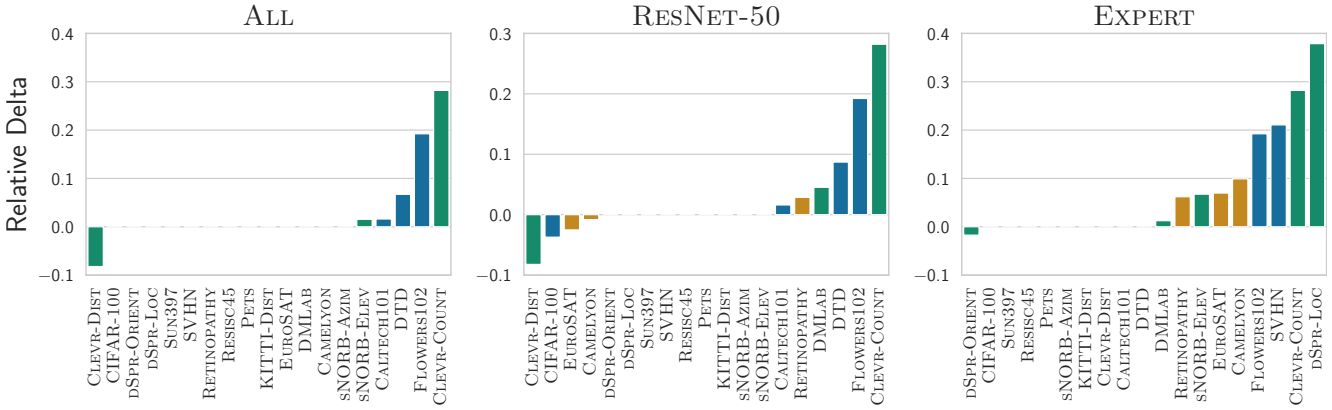

Figure 22. Relative delta between the hybrid Linear (positive if better) and hybrid kNN search strategy (negative if better) for $B = 2$.

F. On the impact of the dimension on kNN

As described in Section 5.4, in this section we show that there is no signification correlation (positive or negative) between the kNN classifier accuracy and the dimension of the representation that it is evaluated on. We see this by running a linear correlation analysis between the dimension of the each representation and the achieved kNN classifier accuracy. In order to have a single point for each possible dimension, and to avoid an over-representation of the expert models, which have all the same dimension, for each dimension we selected the model that achieves the highest kNN accuracy. We do this for all pairs of dimensions and datasets. In Figure 23 we present three hand-picked datasets that achieve (a) the highest anti-correlation value, (b) the lowest absolute correlation value, and (c) the highest correlation value.

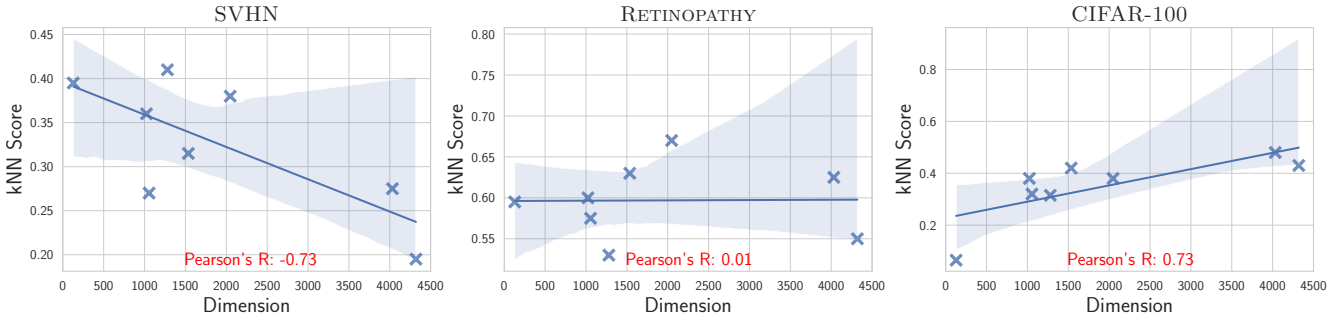

Figure 23. Three examples of datasets in which the analysis of the dimension of the representation compared to the resulting kNN scores results in a negative correlation **(left)**, no correlation at all **(middle)**, and a positive correlation **(right)**.

G. Budget per method

In this section, we report the budget each method requires in order to achieve zero regret per pool. Notice that the strategy "Oracle" refers to the task-agnostic oracle which ranks models based on their achieved average accuracy over all datasets. Even though this is not practical, it enables us to have a task-agnostic method that is able to achieve zero regret eventually as every model (even an expert) is included in this ranking. We split the results by the dataset types and notice that there are some clear patterns between a pool, a dataset type and the required budget for task-aware or task-agnostic methods. For instance, one observes that the linear strategy performs well on all the natural datasets across all the pools except IMNETACCURACIES. Structured and specialized datasets seem to be harder for this proxy task, except for the EXPERT pool. Finally, kNN consistently performs slightly worse than the linear proxy across all pools.

	ALL			RESNET-50			EXPERT		
Dataset	Oracle	Linear	kNN	Oracle	Linear	kNN	Oracle	Linear	kNN
CALTECH101 ●	5	1	4	3	1	2	1	1	1
CIFAR-100 ●	1	2	2	1	2	6	1	1	1
DTD ●	9	2	3	6	2	2	3	2	1
FLOWERS102 ●	26	1	2	20	1	2	11	1	2
PETS ●	13	1	1	9	1	1	5	1	1
SUN397 ●	7	1	1	5	1	1	2	1	1
SVHN ●	1	23	37	2	9	13	15	13	15
CAMELYON ●	23	5	9	17	4	8	9	1	4
EUROSAT ●	1	4	19	14	16	7	4	5	5
RESISC45 ●	1	35	34	3	1	1	2	2	2
RETINOPATHY ●	2	4	8	23	14	9	12	2	7
CLEVR-COUNT ●	7	1	10	5	1	9	2	1	6
CLEVR-DIST ●	44	5	4	35	5	4	10	6	2
DMLAB ●	1	18	29	3	8	4	7	1	6
DSPR-LOC ●	9	25	21	6	19	17	3	3	4
DSPR-ORIENT ●	30	9	21	23	8	19	12	5	12
KITTI-DIST ●	3	8	14	1	7	10	1	3	2
SNORB-AZIM ●	1	45	25	32	3	5	1	2	15
SNORB-ELEV ●	37	1	2	30	1	1	12	1	5

Table 4. Budget required to achieve zero regret per datataset and strategy on the pools ALL, RESNET-50 and EXPERT.

Dataset	DIM2048 Oracle	DIM2048 Linear	DIM2048 kNN	IMNETACCURACIES Oracle	IMNETACCURACIES Linear	IMNETACCURACIES kNN
CALTECH101 ●	3	1	4	3	7	12
CIFAR-100 ●	1	5	10	1	2	2
DTD ●	7	2	3	2	7	9
FLOWERS102 ●	24	1	2	1	14	15
PETS ●	11	1	1	2	2	3
SUN397 ●	5	1	1	1	2	2
SVHN ●	2	12	15	1	14	15
CAMELYON ●	21	5	9	2	7	8
EUROSAT ●	17	21	8	1	2	8
RESISC45 ●	3	1	1	1	14	14
RETINOPATHY ●	6	12	6	2	3	4
CLEVR-COUNT ●	5	1	10	2	12	12
CLEVR-DIST ●	42	5	4	14	6	4
DMLAB ●	9	10	8	1	9	12
DSPR-LOC ●	7	25	21	1	15	13
DSPR-ORIENT ●	28	9	21	1	8	12
KITTI-DIST ●	1	8	13	2	13	15
SNORB-AZIM ●	37	4	7	1	14	9
SNORB-ELEV ●	35	1	2	1	15	15

Table 5. Budget required to achieve zero regret per datataset and strategy on the pools DIM2048 and IMNETACCURACIES.

H. Task2Vec results

As described in Section 2 and 6 of the main part of this work, we report some preliminary results on extending our model search formulation to meta-learned task-aware search strategies. In this section we look at Task2Vec [1], the predominant method in this area. Following the experiments by the authors of Task2Vec, we perform our evaluation in a leave-one-out (LOO) fashion. Concretely, for each dataset in the VTAB benchmark, we run the evaluation by taking the other 18 datasets as the benchmark set of datasets for the meta-learning part of the search strategy. We use the code provided with the Task2Vec paper[3] to calculate the Task2Vec representations on all VTAB datasets with the `ResNet34` probe network and the `variational` method to get the Fisher information matrix (FIM) for each task. In order to determine the nearest task for each dataset in the VTAB benchmark, we use the normalized cosine function, which translates to $d_{sym}(F_a, F_b)$ in the related work. We are not able to use the asymmetric distance function $d_{asym}(t_a \rightarrow t_b)$ proposed by the authors of Task2Vec due to the lack of generalist task in the benchmark set, which would yield the trivial embedding.

Results. We report the nearest task for each VTAB dataset in Table 6. Notice that the nearest tasks for natural and structured datasets often belong to the same category. When analyzing the relative regret from selecting the two best models from the nearest task to fine-tune across different pools in Figure 24, we realize that, unsurprisingly, tasks such as FLOWERS102, where the nearest task seems to not be similar enough, suffer from a high regret of not finding the best expert model. Other tasks, such as EUROSAT, can benefit from picking the two best models instead of only one, a corner case beyond the initial analysis of the Task2Vec paper. Furthermore, we remark a high discrepancy between the regrets for tasks which are symmetrically the nearest tasks of each other (e.g. DSPR-LOC and DSPR-ORIENT). When comparing the relative regret between the task-agnostic selection and the meta-learned task-aware strategy for a fixed budget of $B = 1$, presented in Figure 25, we see that the simple task-agnostic strategy is on par, or better by a small margin, than the computational more demanding approach on most datasets. This findings are supported by the authors of Task2Vec in Figure 3 in [1], where the expert selection procedure is outperformed by the simple generalist approach in 15 out of 50 tasks, sometimes by a large margin. When increasing the budget to $B = 2$ and comparing the hybrid Task2Vec strategy to the Task2Vec strategy, the benefits vanish and hybrid Task2Vec does not yield any significant improvements except for the EXPERT pool (cf. Figure 26).

Limitations and future work. Notice that we only inspect the best model(s) from the single nearest task. For $B > 1$, one could also look at the second nearest task. Preliminary results did not show any improvements for $B = 2$. Extending and designing new search strategies for meta-learned task-aware search and larger budgets is beyond the scope of this work. Furthermore, when exploring this area, one should carefully analyze the impact of having different sets of benchmark dataset. For instance, if a very similar dataset to the user's dataset is in the set of benchmark tasks, naturally the meta-learned task-aware will be very strong. Finally, notice that the asymmetric distance function proposed by the authors of Task2Vec, despite not being applicable to our setting, has strong resembles to the idea of balancing the search between picking a generalist (in their words, falling back to the trivial embedding), and the specialist model originating from the nearest task in the task embedding space. Enabling such a differentiation across all model search strategies without having to increase the budget offers an interesting research question for the future.

[3] https://github.com/awslabs/aws-cv-task2vec

Dataset	Nearest Task
CALTECH101 ●	SUN397 ●
CIFAR-100 ●	EUROSAT ●
DTD ●	SUN397 ●
FLOWERS102 ●	SUN397 ●
PETS ●	CALTECH101 ●
SUN397 ●	CALTECH101 ●
SVHN ●	SNORB-ELEV ●
CAMELYON ●	KITTI-DIST ●
EUROSAT ●	CAMELYON ●
RESISC45 ●	DTD ●
RETINOPATHY ●	SNORB-ELEV ●
CLEVR-COUNT ●	CLEVR-DIST ●
CLEVR-DIST ●	CLEVR-COUNT ●
DMLAB ●	KITTI-DIST ●
DSPR-LOC ●	DSPR-ORIENT ●
DSPR-ORIENT ●	DSPR-LOC ●
KITTI-DIST ●	CAMELYON ●
SNORB-AZIM ●	SNORB-ELEV ●
SNORB-ELEV ●	SNORB-AZIM ●

Table 6. Nearest tasks using Task2Vec for each VTAB dataset.

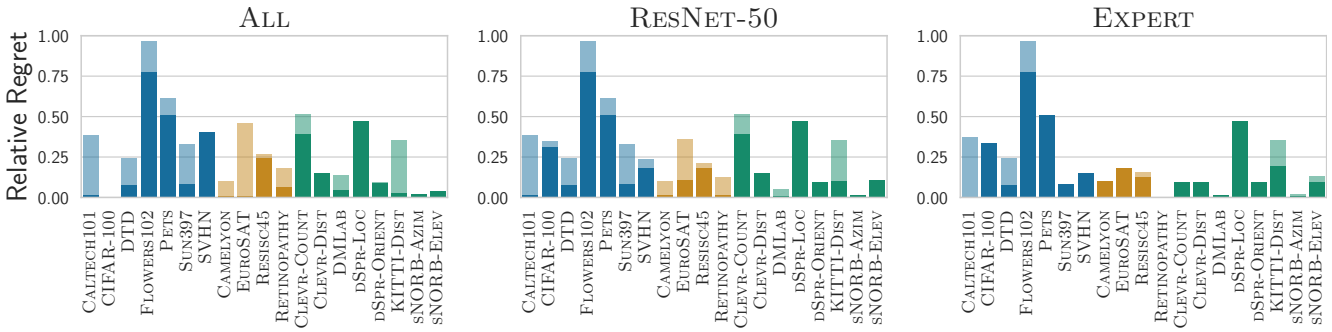

Figure 24. Relative regret for the Task2Vec search strategy with $B = 1$ (transparent) and $B = 2$ (solid).

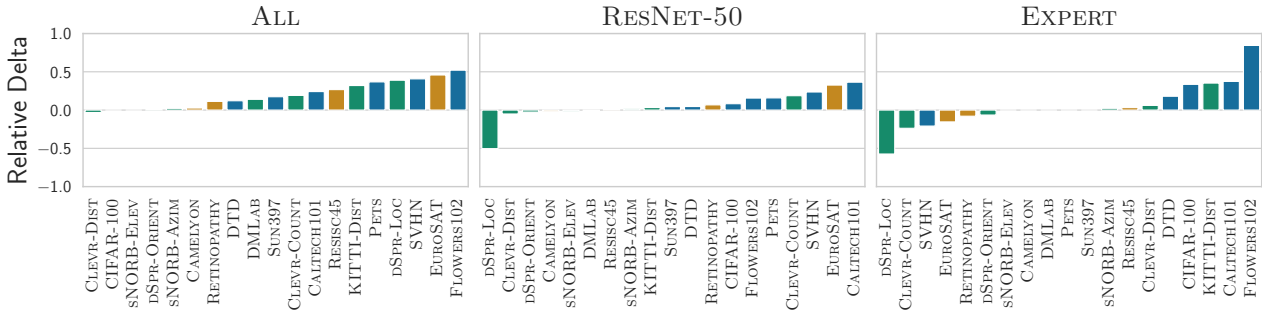

Figure 25. Relative delta between the task-agnostic (positive if better) and the Task2Vec meta-learned task-aware search strategy (negative if better) for $B = 1$.

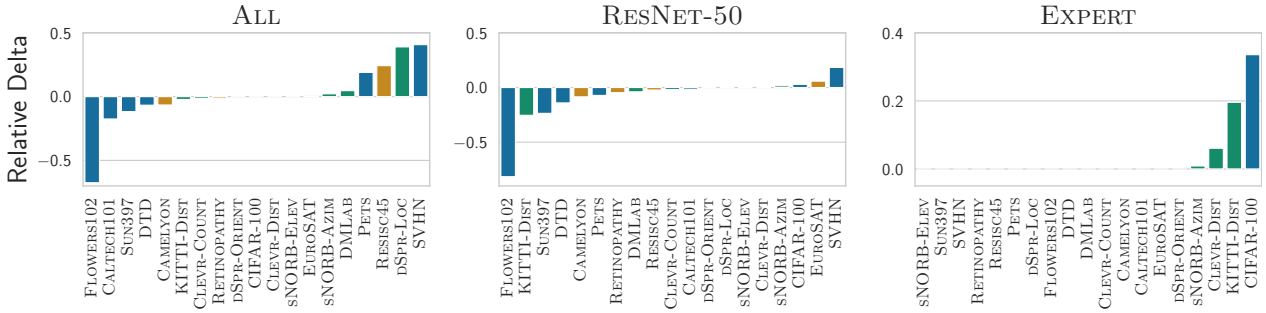

Figure 26. Relative delta between the hybrid Task2Vec (positive if better) and Task2Vec search strategy (negative if better) for $B = 2$.

I. Early stopping as a proxy

As shown in this work, using a linear classifier on top of frozen representation as a proxy for picking which model to fine-tune can yield high regret. As a natural attempt to bypass this limitation, we define a computationally more demanding task-aware proxy method by fine-tuning each network, but only for a very small number of iterations, and use the resulting score as a proxy for performing the model search. We focus on a setting where we stay as close to the downstream training scenario described in Section 2 and 4 as possible. Concretely, we initialize the new head as a zero-valued vector, run SGD for 10 iterations with a batch size of 512 (yielding 5 epochs), and use the largest validation accuracy achieved over the two choices of learning rates (0.1 and 0.01). For robustness, we run this procedure 5 times and take the median validation accuracy. When inspecting the performance of ResNet50 pre-trained architectures, we realize that gradually increasing the learning rate over multiple iterations results in large variance across the different runs. Omitting such a "warm-up" strategy in the few iterations scenario yields more stable results.

Results. The results in Figures 27, 28 and 29, follow the ones using the linear proxy on top of frozen representations. In fact, despite being computationally more demanding, fine-tuning for a few iterations does not bypass the limitations of using the simpler linear classifier proxy in our setting. We believe that the reasons are mainly due to the complex hyper-parameter search one should perform especially in the presence of varying architectures. When having only a few iterations, running any warm-up phase is non-trivial and thus selecting the *best* initial learning rate becomes crucial. Preliminary evaluations showed that limiting the number of fine-tune iterations increases the impact and sensitivity to hyper-parameters. When comparing the validation accuracies between early stopped and fully fine-tuned models, some combination of pre-trained models and dataset require a small initial learning rate to not overshoot in the limited iterations scenario, whereas others would require a larger learning rate in order to be able to increase the accuracy significantly. Further exploring this line of work is beyond the scope of this paper and is left for future work.

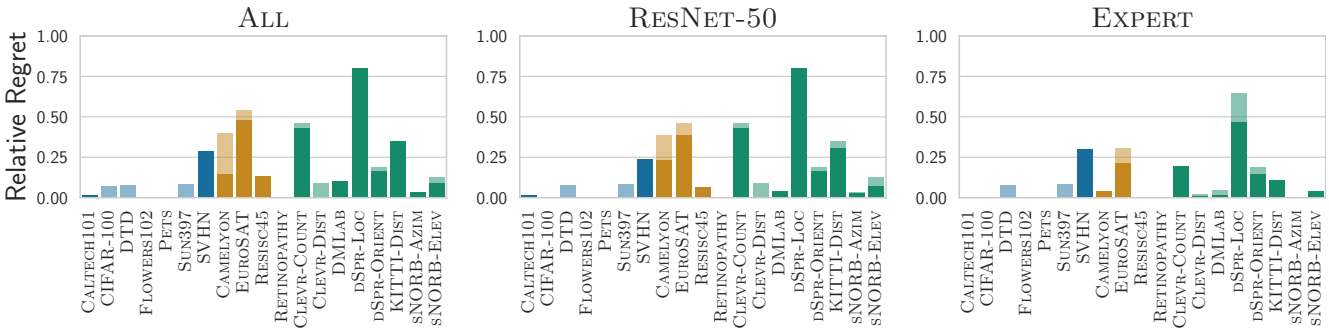

Figure 27. Relative regret for the EarlyStop search strategy with $B = 1$ (transparent) and $B = 2$ (solid).

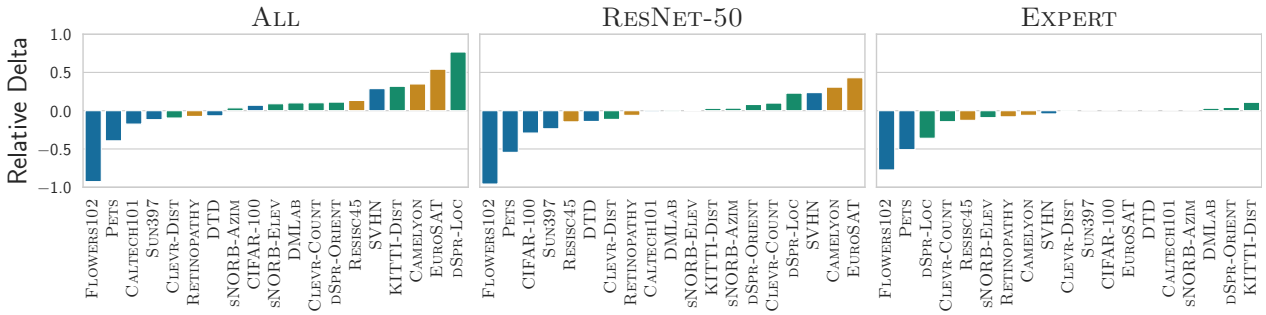

Figure 28. Relative delta between the task-agnostic (positive if better) and the EarlyStop task-aware search strategy (negative if better) for $B = 1$.

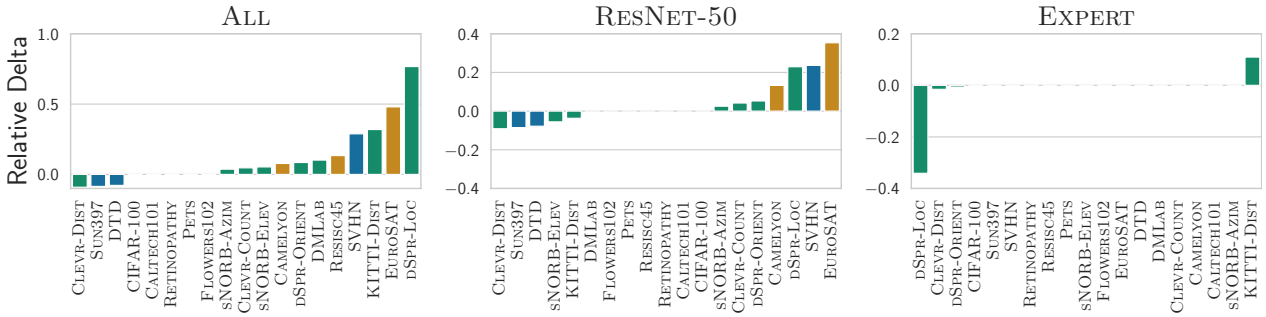

Figure 29. Relative delta between the hybrid EarlyStop (positive if better) and EarlyStop search strategy (negative if better) for $B = 2$.

J. All fine-tune accuracies and picked models

Finally, we provide plots that summarize all the results of the conducted large-scale experiment in a single overview per pool. The plots highlight the range of test accuracies amongst all the fine-tuned models, as well as the returned top-1 models ($B = 1$) for the three strategies – task-agnostic, task-aware linear and task-aware kNN.

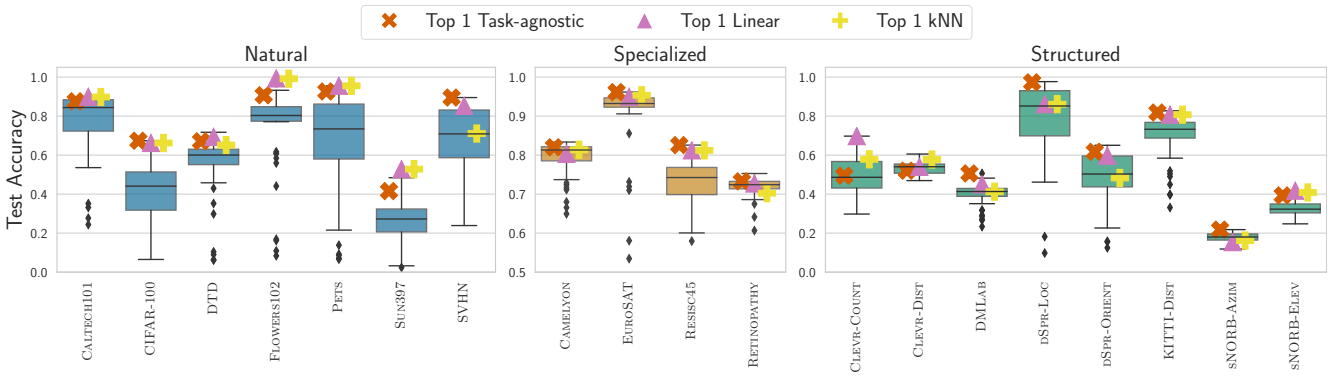

Figure 30. Pool ALL.

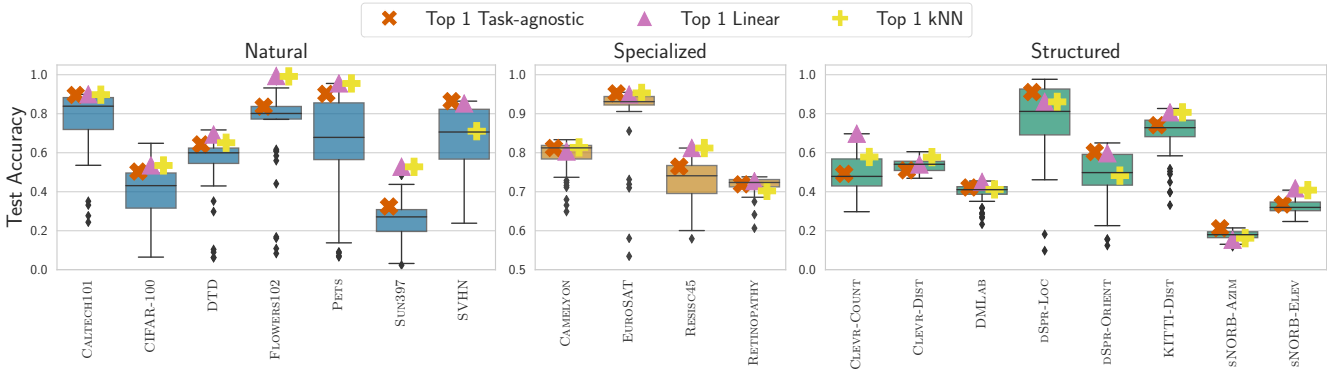

Figure 31. Pool DIM2048.

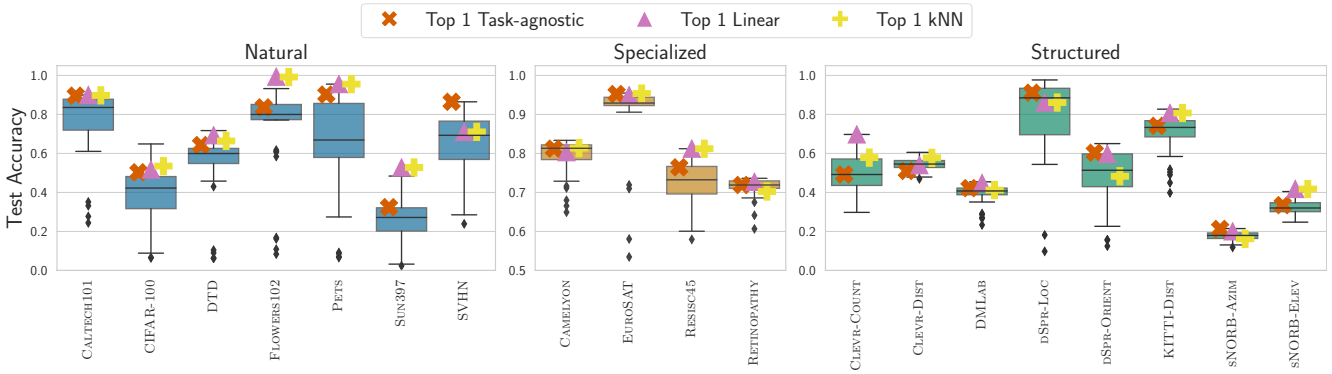

Figure 32. Pool RESNET-50.

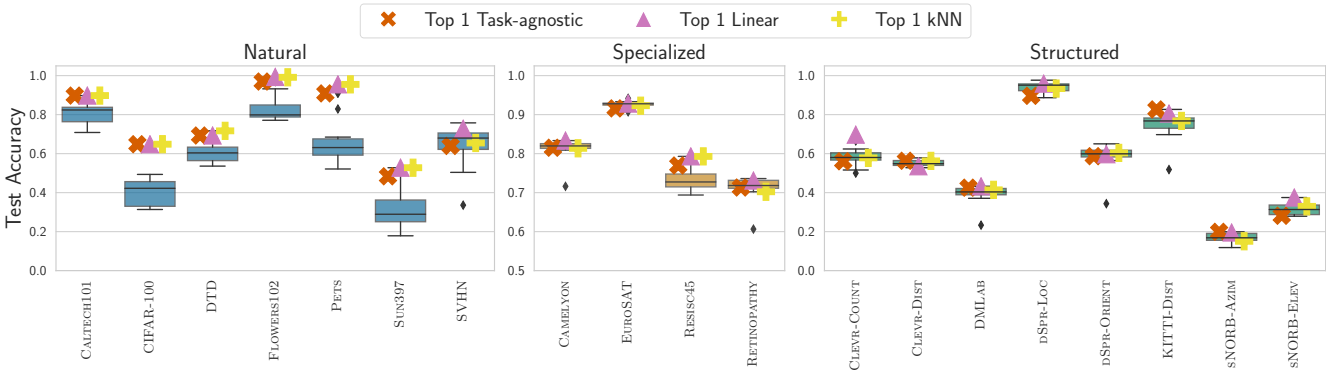

Figure 33. Pool EXPERT.

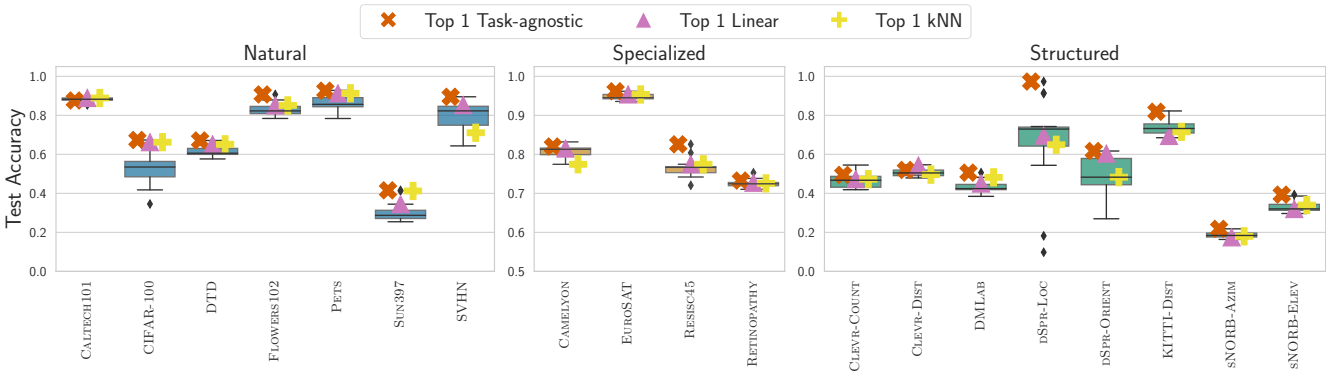

Figure 34. Pool IMNETACCURACIES.