A EXPERT PREDICTOR NETWORKS

An expert predictor network (EPN) tries to directly predict the relevant expert for an input image, using only the image itself as input. We first train the EPN upstream, and then we apply it downstream to select the most relevant expert for a new task by aggregating its output on all the images in the task.

EPN Upstream Training. As we described in section 4, we have split the upstream dataset D_U into a collection of subsets $\{D_e : 1 \leq e \leq E\}$, with $D_e = (X_e, Y_e) \subseteq D_U$. In order to train the EPN, we simply assign the expert identity e as the label of all images in X_e, and train the network in a supervised manner (using softmax cross-entropy) to predict the expert. Expert slices $\{D_e\}$ are not disjoint, thus, it is possible that an individual image appears multiple times in the training data for the EPN with different expert identities. Because subsets sizes are different, and in order not to favor any particular expert, we resample the training images so that each expert is seen equally often. Intuitively, this classification problem should be substantially easier than predicting the upstream classes y directly, as there are much fewer experts than upstream classes.

Downstream Expert Selection. Suppose we are given a downstream task, containing images $X_T = \{\mathbf{x}_1, \ldots, \mathbf{x}_{N_T}\}$. We first apply a forward pass on X_T using the EPN. Let $Q_{\text{EPN}}(e \mid X = \mathbf{x}_i)$ be the probability assigned by the EPN to expert e for input image $\mathbf{x}_i$. In order to make a single decision for the whole downstream task, we combine those probabilities using a log-linear transformation.

We select the expert as follows:

$$\hat{e} = \arg\max_e \frac{1}{N_T} \sum_{i=1}^{N_T} \log Q_{\text{EPN}}(e \mid X = \mathbf{x}_i). \tag{1}$$

The log-linear combination of per-example probabilities was obtained after several experiments with a number of functions. Intuitively, this transformation penalizes experts that only apply to a subset of the downstream data, but are not relevant to other downstream examples.

A major drawback of the EPN is the fact that it does *not* use or benefit from the downstream labels. Imagine there is an image dataset with pictures containing simultaneously both lions and elephants. Suppose we are faced with two different downstream tasks based on the same inputs: one is to count lions, the other is to count elephants. Furthermore, imagine our experts happen to include *lion* and *elephant*. Depending on the task, it would be reasonable to choose one or the other expert. Unfortunately, the basic EPN approach is agnostic to the outputs, and –as the input images are identical– it would return the same selected expert in both cases.

B KULLBACK–LEIBLER DIVERGENCE

Since our expert datasets $\mathbf{D}_e$ were built based on the hierarchy of labels in the upstream dataset, it is reasonable to assume that the prior distribution of the labels in each $\mathbf{D}_e$ differ across experts e. Let P_e be this prior distribution for expert e. Then, we can use a divergence measure, such as the Kullback–Leibler (KL), to determine which expert to use. If one assumes that the downstream dataset is well represented by the upstream dataset $\mathbf{D}_U$ (although not necessarily by an individual $\mathbf{D}_e$), one can use the baseline neural network $\mathbf{B}$ to approximate the distribution of *upstream* labels conditioned to the set of downstream images:

$$Q(Y) \coloneqq \frac{1}{N_T} \sum_{j=1}^{N_T} Q_{\mathbf{B}}(Y \mid X = \mathbf{x}_j), \tag{2}$$

where $Q_{\mathbf{B}}(Y \mid X = \mathbf{x}_j)$ is the probability distribution given by $\mathbf{B}$ over each image in the set $X_T = \{\mathbf{x}_1, \ldots, \mathbf{x}_{N_T}\}$ of downstream images. Then, we simply select the expert with the lowest KL divergence:

$$\hat{e} \coloneqq \arg\min_e D_{\text{KL}}(P_e \parallel Q) \tag{3}$$

This allows us to leverage the baseline model that we already trained, and not train an auxiliary neural network to predict the expert to use, like the EPN in appendix A does. In addition, this has the benefit

of using information about the distribution of upstream classes, which may be useful when the target classes are well represented among the upstream ones.

In our case, the upstream datasets consists of multi-labeled images. The distribution of labels given to a particular image is modelled by the neural network as a joint distribution of independent Bernoulli random variables. Assuming this independence also holds for P_e, one can then compute eq. (3) very efficiently. Of course, this assumption is not true in either case (e.g. the presence/absence of the *dog* and *animal* labels is not independent), but it is standard practice for multi-label classification.

C Further Results on k-Nearest Neighbors

In this section, we present the kNN accuracy distribution per dataset that we found for both JFT and ImageNet21k experts. A flat curve indicates differences across experts may not be very relevant for the downstream task, while steep regions suggest strong decreases in value among expert models.

C.1 kNN Hyperparameters

We select the expert to transfer using the kNN transfer proxy with $k = 1$ and a Euclidean distance metric. We use 1 000 training examples in all datasets (including those in the comparison with Domain Adaptive Transfer Ngiam et al. (2018)), and compute kNN leave-one-out cross-validation to compute the accuracy per expert. Finally, we select the one with highest accuracy. For VTAB-1k this corresponds to the entire training set per task; whereas for the other tasks, we randomly sample 1k training examples. We do not perform special data pre-processing, and simply resize and crop to 224×224, as done in upstream evaluation.

We used a NVDIA V100 GPU to perform the kNN selection for each dataset, with this hardware, selecting among 240 models takes less than 2 hours.

C.2 Architecture Comparisons

In this section we look at the kNN accuracy *before* transferring the experts, and –in particular– at how it depends on the architecture choice. Recall each expert is associated with one slice of the upstream data. For any given slice, we have trained both a full ResNet50 network, and adapters attached to a pretrained ResNet50. We plot the accuracy achieved by these representations (scatter-plot, full at x-axis; adapters at y-axis) for all experts for each group of VTAB datasets. We look at JFT experts (Figure 4) and at ImageNet21k experts (Figure 5). Ideally, we would expect some positive correlation if expert representations were somewhat similar regardless of the architecture.

JFT Experts. The first seven plots in Figure 4 show the results in natural datasets. While most experts do not seem relevant –and performance seems a bit uncorrelated between both types of models–, in most datasets we see that there are a few good experts (top right corner) which offer the strongest performance despite of the selected architecture. SVHN seems to be an exception.

Similar plots are displayed in the following 4 and the last 8 plots in Figure 4 for specialized and structured datasets, respectively. Few datasets show agreement on the most promising expert slices, such as Eurosat or Resisc45. Unfortunately, there is no clear agreement in most specialized and structured datasets.

ImageNet21k Experts. We see a reasonable agreement among the best ImageNet21k experts in natural datasets (see first 7 plots in Figure 5). While not as correlated as in the case of natural datasets, we still see some positive relationship in some specialized (Eurosat, Resisc45, Patch Camelyon) and structured (Clevr Count, DSprites Position, Smallnorb Azimuth) datasets.

C.3 kNN Accuracy Distribution for JFT Experts

Figure 6 shows the distribution of the kNN accuracy obtained from the embedding of each of the experts trained on JFT. In each case (full and adapters), the kNN accuracy of the 240 experts has been sorted in a decreasing manner. Note that we pick the single expert with highest-score (although other approaches are possible).

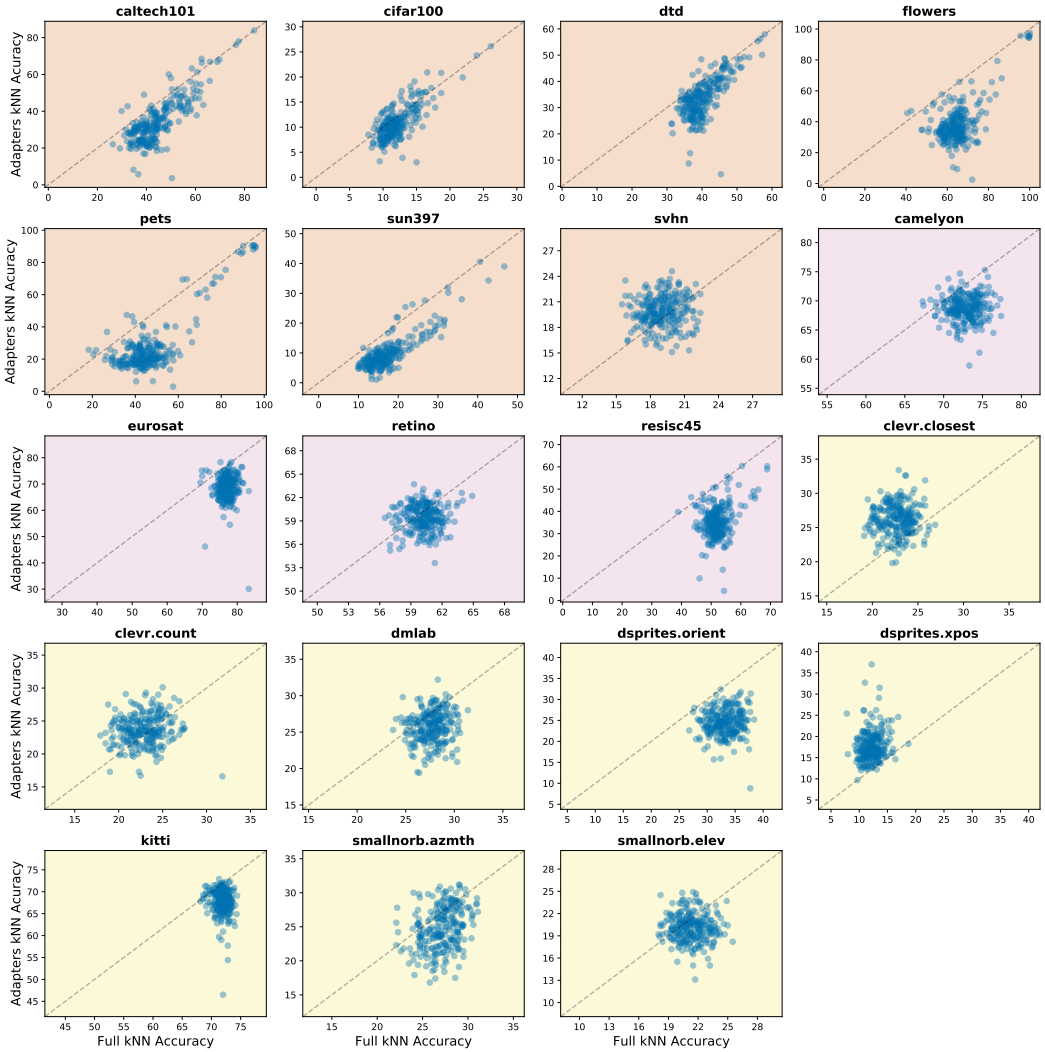


Figure 4: **JFT Experts**. Each point is one expert (upstream data slice); the x-axis represents the kNN accuracy before downstream finetuning of the expert trained on a *full* network. The y-axis displays the kNN accuracy before downstream finetuning of the expert trained with an *adapter* module. The background color indicates the dataset group: ● NATURAL, ● SPECIALIZED, and ● STRUCTURED. There are 240 experts. Identity dashed line shown too.

In most ● NATURAL datasets, we observe that full JFT experts are on average better than their adapter counter-parts. In datasets like Caltech101, Cifar100, or DTD, it seems these differences do not affect the top experts, while in others (such as Flowers, Pets, and Sun397) the differences still apply to the best expert. Also, overall, we see that in natural datasets there are usually *strong* differences between good and bad experts. The range of kNN accuracies is pretty large for Caltech101, Flowers, or Pets, where some experts seem to already solve the task, while others lead to quite poor accuracies. The latter may be fixed to some extent by downstream fine-tuning.

In the ● SPECIALIZED group we see a similar pattern in the comparison between full and adapter-based experts. However, the accuracy range of variations (except, maybe, at the very worst end) is narrower.

The story for ● STRUCTURED datasets with JFT experts is a bit different. In some datasets, adapters models lead on average to better initial representations (such as Clevr Closest and Clevr Count, or dSprites Position). As with structured datasets, the difference between the best and the worst experts is shorter. This may be in part explained by the hardness of the task itself (the average accuracy *after*

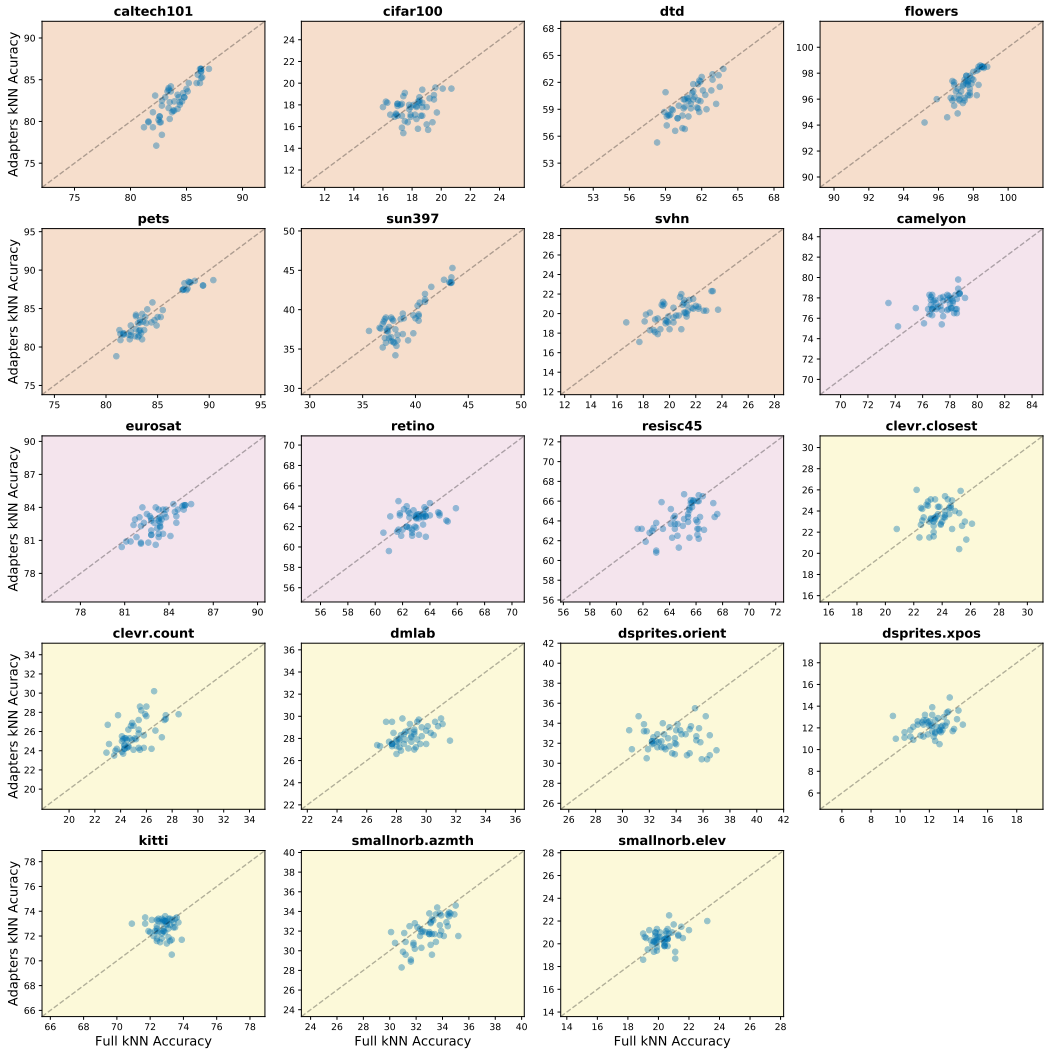


Figure 5: **ImageNet21k Experts**. Each point is one expert (upstream data slice); the x-axis represents the kNN accuracy before downstream finetuning of the expert trained on a *full* network. The y-axis displays the kNN accuracy before downstream finetuning of the expert trained with an *adapter* module. The background color indicates the dataset group: ● NATURAL, ● SPECIALIZED, and ● STRUCTURED. There are 50 experts. Identity dashed line shown too.

fine-tuning is definitely lower than in the natural case), but there are some counter-examples to this, like dSprites Position where final accuracies go up to around 90%.

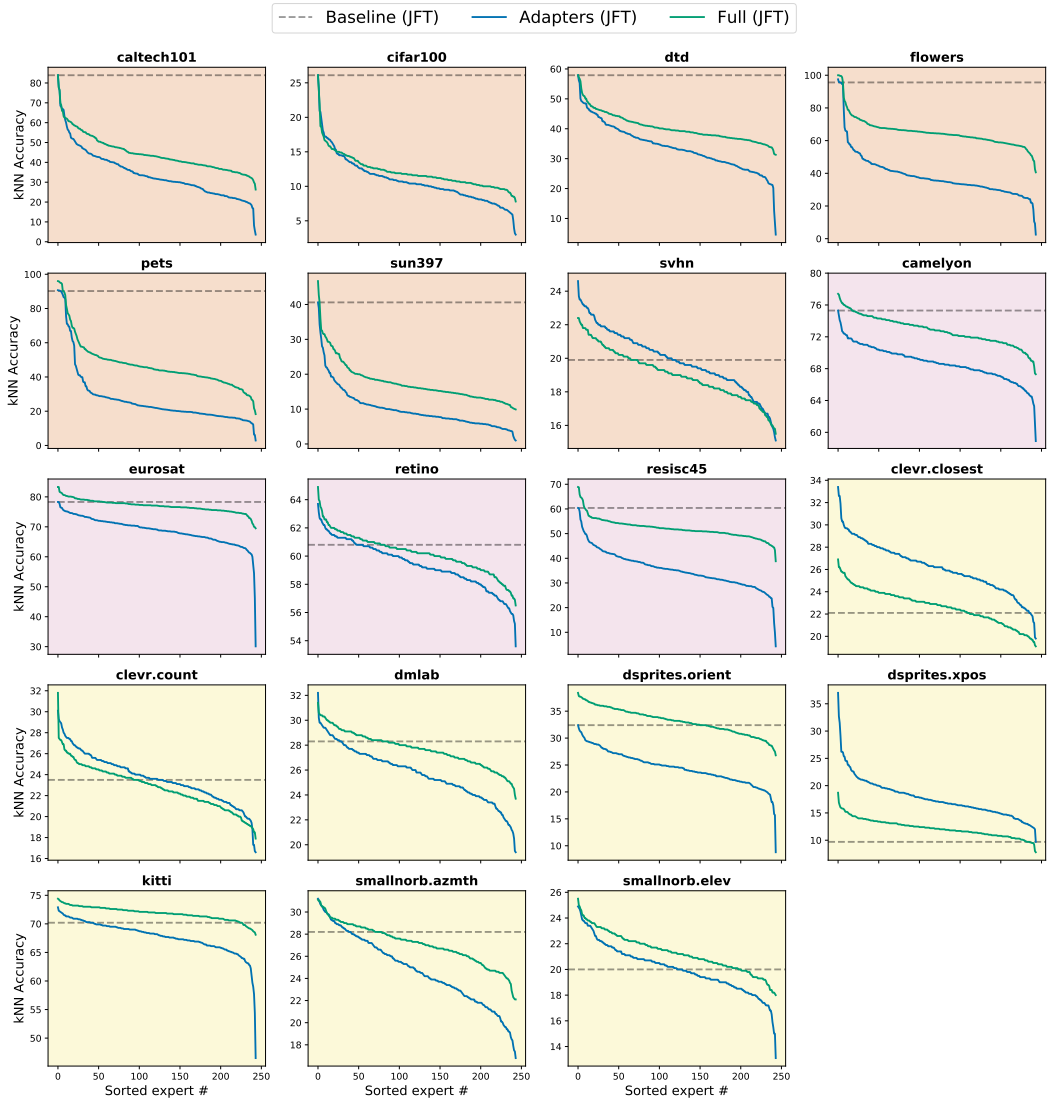


Figure 6: Distribution of the kNN accuracy from experts trained on JFT. The dashed lines shows the kNN accuracy of the baseline model. In each dataset, the experts are sorted according to their accuracy. The background color of the plot represents the group of the dataset: ● NATURAL; ● SPECIALIZED; ● STRUCTURED.

C.4 kNN ACCURACY DISTRIBUTION IMAGENET21K EXPERTS

Figure 7 presents the distribution of the kNN accuracy obtained from the embedding of each of the experts trained on the ImageNet21k dataset. For context, in all the plots we also show the Top-50 JFT full experts.

For ● NATURAL tasks, we observe that the quality of the ImageNet21k expert representations is way more homogeneous than the JFT one. Accordingly, finding the right expert in JFT may be more important (as accuracy decreases fast), while it may provide even more target-tailored representations (see Oxford Flowers and Oxford Pets). Overall, ImageNet21k accuracies seem more stable, and differences between full and adapters are modest.

Overall, both full and adapter ImageNet21k experts seem to perform similarly on ● SPECIALIZED tasks. The plots suggest that ImageNet21k experts are a bit ahead of the full JFT ones (even though the gap at the top tends to close).

We see a few distinct behaviors in ● STRUCTURED. There tend not to be very remarkable winner experts, and full experts may provide a small boost compared to adapter-based ones. In most datasets, the Top-50 full JFT experts outperform the ImageNet21k ones.

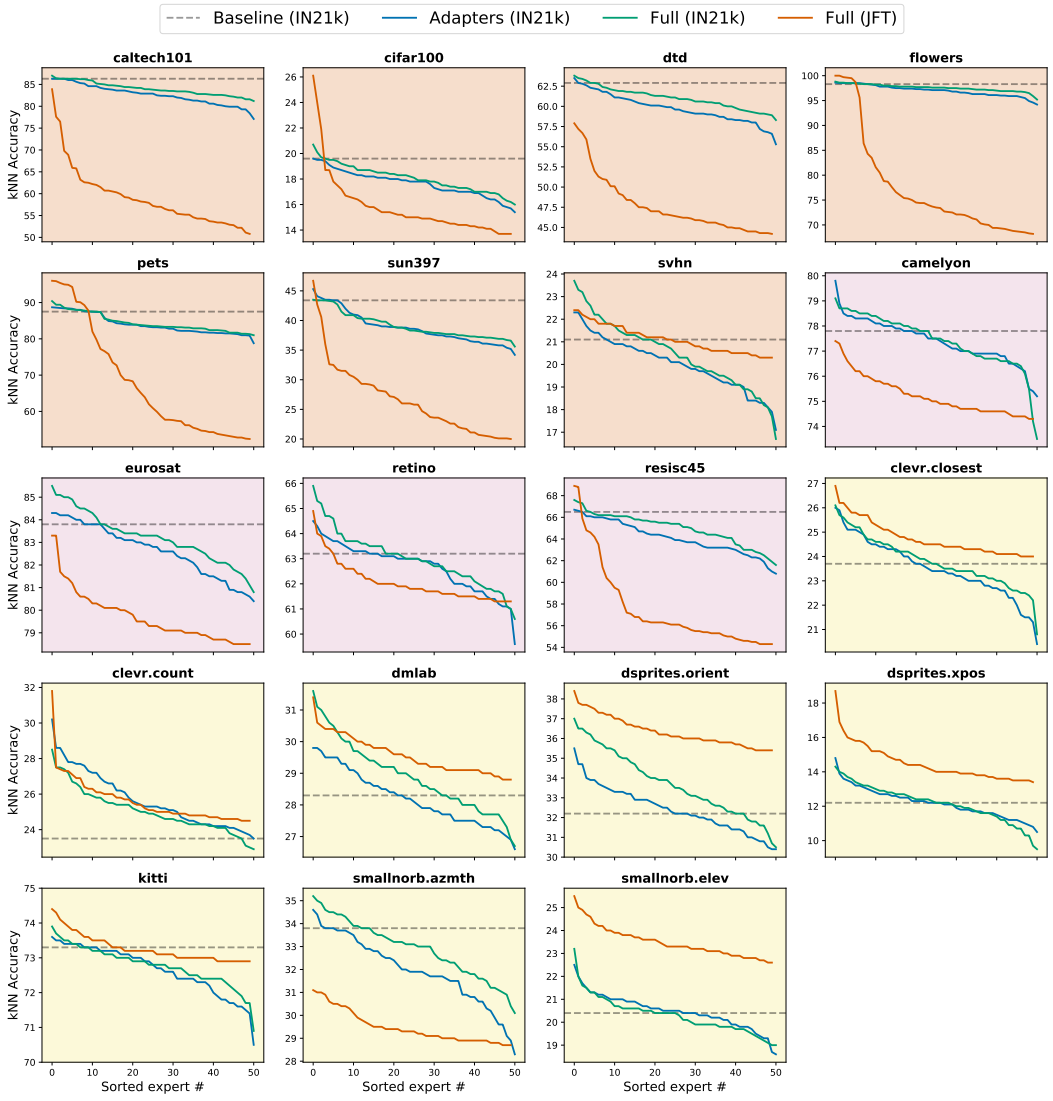


Figure 7: Distribution of the kNN accuracy from experts trained on ImageNet21k. The dashed lines shows the kNN accuracy of the baseline model. The performance of the top-50 JFT full experts on each dataset is also shown. In each dataset, the experts are sorted according to their accuracy. The background color of the plot represents the group of the dataset: ● NATURAL; ● SPECIALIZED; ● STRUCTURED.

C.5 kNN Accuracy Distribution for Consecutive Checkpoints of ImageNet21k Baseline

In this subsection, we study how representations evolve during training. In order to do that, we stored 157 checkpoints –equally spaced– over the training of our ImageNet21k baseline. We trained the model for 90 epochs. For each dataset, we compute the kNN accuracy of the checkpoints, and display the curves in fig. 8. As an auxiliary line, we also show the mean kNN accuracy across all the checkpoints.

There are some clear differences depending on the *type* of dataset. In the case of ● Natural images, it seems that more training leads to better representations. The kNN accuracy tends to increase (Cifar100 and SVHN are exceptions). ● Specialized datasets behave in a different way; while there is an initial boost in accuracy (i.e. trained models are better than randomly initialized ones), long training only leads to very minor improvements in representation quality for these tasks. Finally, ● Structured datasets have extremely flat footprints. This probably means that our semantic experts are not a good fit for this type of task.

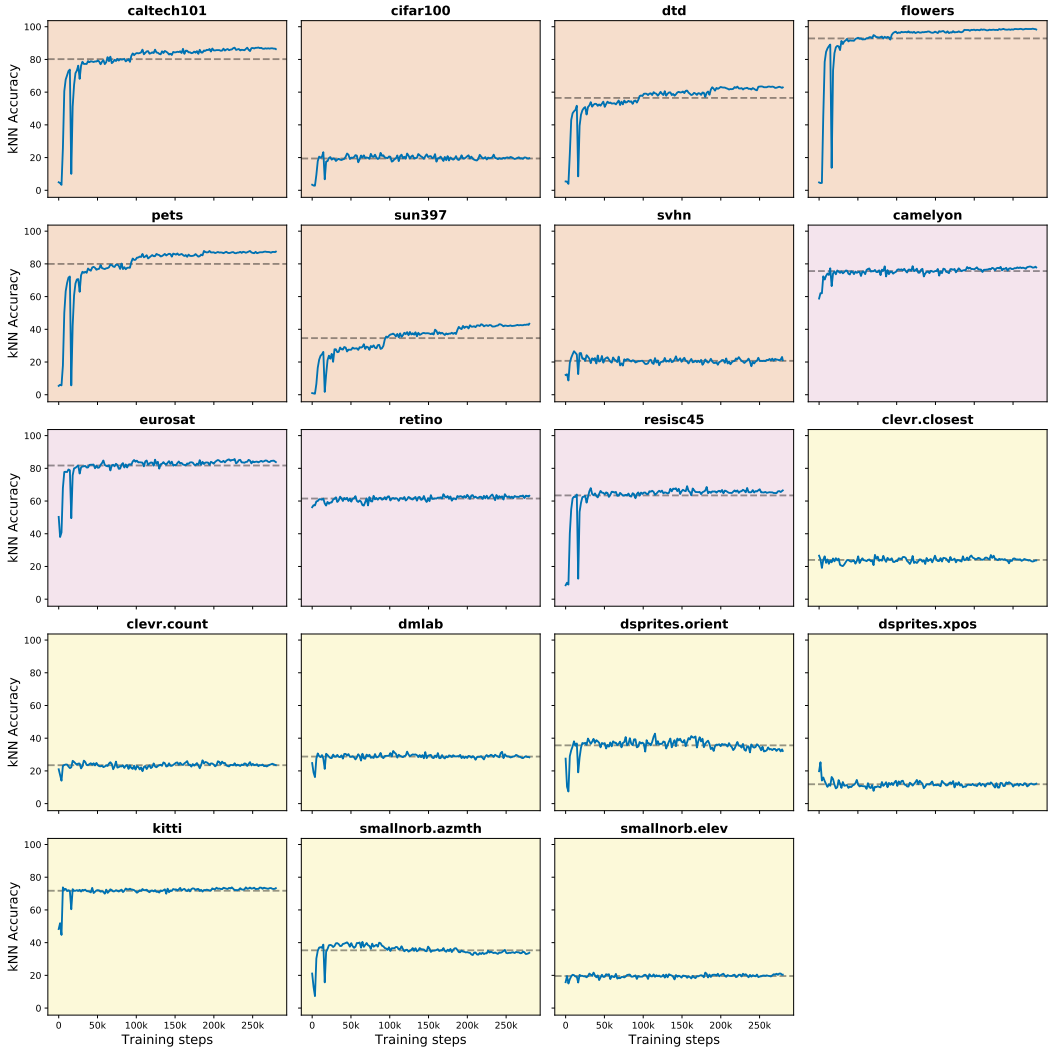


Figure 8: Accuracy of kNN using consecutive checkpoints stored during the ImageNet21k baseline training (90 epochs). The dashed line represents the mean value across all checkpoints and it is useful to point out lack of improvement over time in some cases. The different types of datasets are highlighted by the background color: ● Natural, ● Specialized and ● Structured.

C.6 SELECTED EXPERTS

The following table presents the experts selected by kNN in each of the individual datasets of the ● NATURAL, ● SPECIALIZED and ● STRUCTURED groups.

Table 4: Selected experts by kNN using different expert architectures, in each of the VTAB-1k datasets. Datasets are grouped by ● NATURAL, ● SPECIALIZED and ● STRUCTURED.

Dataset	Full-JFT	Adapter-JFT	Full-INet	Adapter-INet
● Caltech101	*Baseline*	*Baseline*	Physical Entity	*Baseline*
● Cifar100	*Baseline*	*Baseline*	Object	*Baseline*
● DTD	*Baseline*	*Baseline*	Artifact	Artifact
● Oxford Flowers	Plant	Flower	Organism	Physical Entity
● Oxford Pets	Mammal	Carnivore	Carnivore	Carnivore
● Sun397	Structure	*Baseline*	Structure	Structure
● SVHN	Textile	Staple food	Implement	Artifact
● Diabetic Retinopathy	Paper	Material	Food	Plant
● Eurosat	Snow	*Baseline*	Whole	Breathe
● Patch Camelyon	Tree	*Baseline*	Whole	Woody Plant
● Resisc45	Geographical feature	*Baseline*	Instrument	Arthropod
● Clevr Closest	Mode of transport	Canis	Mammal	Relation
● Clevr Count	Snow	Adventure	Spermatophyte	Flower
● DMLab	Sports equipment	Art	Implement	Vertebrate
● dSprites Orientation	Flowering plant	Mode of transport	Clothing	Implement
● dSprites Position	Dish	Toyota	Matter	Chordate
● Kitti	Shoe	Geographical feature	Plant	Abstraction
● Smallnorb Azimuth	Home and garden	Food	Abstraction	Device
● Smallnorb Elevation	Bag	Artwork	Carnivore	Mammal

D UPSTREAM TRAINING

D.1 UPSTREAM TRAINING DETAILS

Unconditional pre-training. We pre-train generic JFT and ImageNet21k models using a similar protocol to the one described in Kolesnikov et al. (2019). In particular, we use SGD with momentum of 0.9, with a batch size of 4096, an initial learning rate of 0.03 (scaled by a factor of $\frac{\text{batch size}}{256}$), and weight decay of 0.001. The JFT backbone model is trained for a total of 30 epochs, while the ImageNet21k is trained for 90 epochs. In both cases we perform training warm-up during the first 5 000 steps by linearly increasing learning rate, and then decay the learning rate by a factor of 10 at $\{\frac{1}{3}, \frac{2}{3}, \frac{5}{6}\}$ of the total duration. During this phase we used a Cloud TPUv3-512 to train each of the baseline models, which takes about 25 hours in the case of JFT and .

Experts training. In order to obtain the expert models, we then further tune these baselines (adding the residual adapters, when applicable) on different subsets of the original upstream dataset. We use a similar setting to the one described before, although we train for much shorter times, use a batch size of 1 024, and use different learning rates. In particular, the full experts use an initial learning rate 10^{-4}, since all the parameters were pre-trained in the earlier phase. The experts with adapters use a larger learning rate of 10^{-1}, as these components are trained from scratch and are the only ones that are tuned. We use the same learning scaling factor and decay schedule as in the previous step. The initial learning rate in each case was decided based on average upstream performance across the different expert datasets. We fine-tune the full experts for 2 epochs, and the adapters for 4 epochs, relative to the size of the entire dataset. The only exception is for the results reported in the comparison with Ngiam et al. (2018), for which we observed in the validation data that full experts trained for 4 epochs performed better.

In both stages, we perform standard data augmentation during training, which includes random image cropping as in Szegedy et al. (2015), random horizontal mirroring, and finally image resize to a fixed resolution of 224×224. When we need to evaluate these models on upstream data (i.e. upstream learning rate selection), we simply resize and crop the images to a fixed resolution of 224×224. Pixel values are converted to the $[-1, 1]$ range.

During this phase we used a Cloud TPUv3-32 to train each of the experts. Training one of the JFT experts for 2 epochs takes about 11 hours, while this is reduced to 30 minutes in the case of the ImageNet21k experts.

D.2 Upstream Freezing

Note that many expert datasets $\mathbf{D}_e$ do not contain any instances of some original upstream classes (for example, the data for the expert *elephant* may not contain any image with the label *vehicle*). As the head is frozen and shared among all experts, the adapters need to find other ways to ignore classes that do not apply at all to the expert. We found this to be beneficial in practice, as we avoided too much upstream dependence on the head (which is later discarded in the transfer stage).

E Visual Task Adaptation Benchmark details

E.1 Classes per Task

The number of classes in the VTAB tasks varies significantly, see Figure 9. As we are most interested in the low-data regime, we fix the number of downstream examples to 1 000, implying that some downstream datasets only contain 3-10 examples per class –like Sun397 or Caltech.

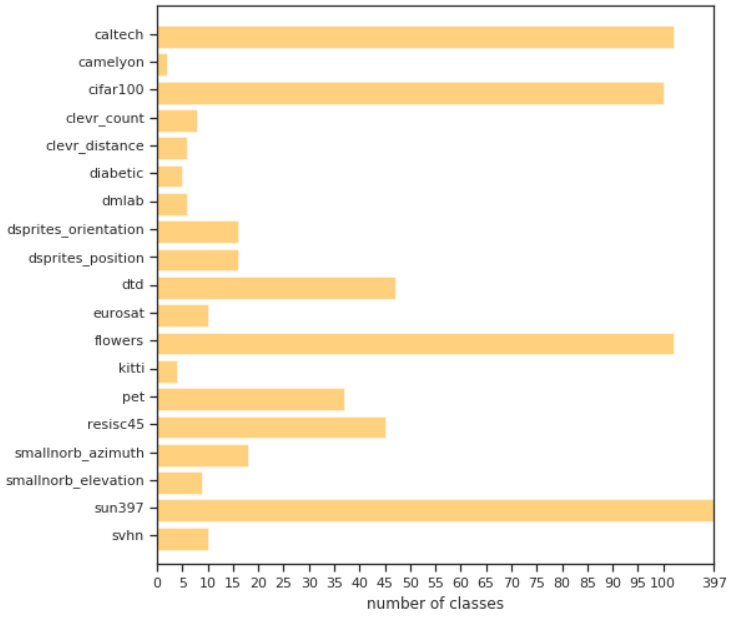


Figure 9: Number of classes per Downstream Task in VTAB.

E.2 Downstream Hyperparameters on VTAB

We use SGD with momentum of 0.9 and a batch size of 512. The initial learning rate is scaled by $\frac{\text{batch size}}{256}$. We don't perform any data augmentation, and resize all the images to a fixed resolution of 224×224. Pixel values are converted to the $[-1, 1]$ range. We perform a restricted hyperparameter search, in particular we follow the *lightweight* suggestion from Zhai et al. (2019). The sweep tries in total 4 different sets of hyperparameters for each dataset.

- Initial learning rate: the values of $\{0.1, 0.01\}$.
- Training schedule: we try with a total training duration of $2\,500, 10\,000$ steps, with a linear warm up of the scaling rate of 200 and 500 steps, respectively. For both durations, the learning rate is reduced by a factor of 10 after $\{\frac{1}{3}, \frac{2}{3}, \frac{9}{10}\}$ of the total number of steps.

Note that the hyperparameter sweep is done by training the models on 800 examples (out of the 1 000 available), and selecting over 200 validation examples. Then, the best combination of hyperparameters is used to re-train the models on the full 1 000 data points.

Because the variability due to random initialization with so few data points is large in some datasets, we perform 10 independent runs of hyperparameter selection, and then re-training the models 3 times for each of the 10 selected hyperparameters. This yields a total of 30 outcomes for each of the VTAB-1k datasets. For each dataset, we report the median over these 30 trials and compute (percentile) bootstrapped confidence intervals at 95% level.

For downstream training we use a Cloud TPUv3-16. In VTAB-1k, the running time depends on the number of steps. It takes 12 minutes to fine-tune one of our experts for 10 000 steps (the longest duration), and just 3 minutes for the shorter schedule of 2 500 steps.

E.3 Additional Results of Different Expert Selection Strategies

In section 6.4, and more precisely in table 1, we studied the performance of random transfer (expert selection uniformly at random). There, we only reported the results corresponding to full experts trained on JFT. For completeness, table 5 shows also the results with adapters. The pattern is fairly similar: random transfer leads to massive losses in Natural datasets (we obtain an almost 35% improvement by applying kNN with respect to random transfer). Domain Prediction and Label Matching also heavily help in these settings. In Specialized and Structured tasks, both Domain Prediction and Label Matching seem to offer little-to-no gains, whereas kNN still leads to a modest boost on Structured, and a decent one on Specialized.

Overall (last column of the table), the improvement is significant and strong for all routing methods.

Table 5: VTAB-1k results of different selection algorithms, using full and adapter experts trained on JFT. The average accuracy across each group of tasks and across all VTAB is reported. In each dataset, the median accuracy over 30 runs is used. Bootstrapped confidence intervals at 95% level.

	Natural	Specialized	Structured	All
Adapters				
Random	58.6 [56.1–59.6]	78.3 [76.8–79.2]	58.6 [57.8–59.6]	62.8 [61.7–63.3]
Domain Prediction	70.8 [69.3–71.6]	75.5 [63.7–78.0]	59.7 [58.2–61.0]	67.1 [64.5–67.9]
Label Matching	75.3 [75.1–75.4]	80.5 [78.2–81.3]	56.1 [51.8–57.0]	68.3 [66.4–68.7]
Performance Proxy	79.0 [78.6–79.1]	81.3 [79.2–82.5]	59.1 [58.3–60.1]	71.1 [70.5–71.6]
Full				
Random	60.6 [59.1–63.9]	81.22 [80.9–81.8]	56.8 [54.9–57.8]	63.3 [62.3–64.6]
Domain Prediction	75.9 [74.4–77.4]	81.5 [81.3–82.2]	**57.0** [56.1–57.4]	69.1 [68.4–69.8]
Label Matching	78.0 [77.8–78.1]	80.3 [79.1–82.5]	56.9 [55.6–57.2]	69.6 [68.9–70.0]
Performance Proxy	**79.7** [79.5–80.0]	**83.6** [83.3–83.8]	55.3 [52.1–56.3]	**70.2** [68.9–70.6]

E.4 Per-Task Results

All VTAB results presented so far were averaged over dataset types (natural, specialized, and structured). In this subsection, we break down the outcomes per dataset. Table 6 shows the mean accuracy (and confidence intervals) for 13 algorithms and 19 datasets. The datasets are sorted according to the data type. The table can be used for reference. Some datasets showcase a wide range of outcomes for any fixed algorithm. In order to expose this in a clear fashion, we present in fig. 10 the individual trial accuracies for the best algorithms and baselines, in all of the VTAB datasets. The fine-tuning process on datasets like Clver Count, dSprites xPosition, or SVHN definitely shows a high variance of test accuracies. The median estimator partially mitigates this effect.

Table 6: Accuracy on the individual datasets of the VTAB-1k benchmark. Algorithms include experts trained on both JFT and ImageNet21k, with adapters and full architectures, and by means of different selection methods (Expert Predictor Network, EPN; Kullback–Leibler, KL; and kNN). In each dataset, the median accuracy over 30 runs is used. Bootstrapped confidence intervals at 95% are shown. Color indicates dataset group: ● NATURAL; ● SPECIALIZED; ● STRUCTURED.

	caltech101	cifar100	dtd	flowers	pets	sun397	svhn	camelyon	eurosat	retino	resisc45	clevr.closest	clevr.count	dmlab	dsprites.orient	dsprites.xpos	kitti	smallnorb.azmth	smallnorb.elev
JFT																			
Baseline	91.7 [91.5–91.8]	68.6 [68.3–68.7]	72.1 [72.0–72.2]	97.2 [97.1–97.2]	91.5 [91.4–91.5]	49.9 [49.9–50.0]	71.2 [70.5–72.0]	81.6 [81.4–83.1]	93.0 [92.9–93.1]	70.0 [69.6–70.2]	81.8 [81.5–81.9]	54.9 [54.5–55.8]	62.8 [60.7–68.0]	45.1 [45.0–45.3]	61.6 [61.1–62.1]	94.9 [93.7–96.2]	79.8 [43.9–80.8]	25.1 [22.0–30.5]	33.6 [33.1–35.5]
Adapters (EPN)	91.7 [91.6–91.7]	34.0 [32.2–34.3]	58.3 [58.0–58.6]	98.2 [98.1–98.2]	91.4 [91.3–91.5]	48.3 [48.1–48.4]	73.7 [63.3–79.2]	79.6 [71.7–83.1]	93.1 [92.2–93.6]	60.0 [53.7–66.4]	69.3 [23.3–74.3]	60.6 [57.3–61.9]	63.3 [56.7–73.3]	45.1 [43.7–46.0]	59.3 [56.9–59.5]	95.8 [95.0–96.8]	79.2 [73.7–80.4]	32.6 [32.1–33.1]	41.8 [37.7–42.6]
Adapters (KL)	91.4 [91.1–91.6]	68.3 [68.2–68.3]	57.5 [57.0–57.7]	98.1 [97.8–98.2]	92.0 [91.9–92.0]	48.3 [48.3–48.4]	71.6 [70.9–72.1]	83.0 [74.1–83.6]	93.0 [92.9–93.1]	68.2 [64.5–70.7]	77.7 [77.5–79.7]	52.2 [49.3–53.8]	59.9 [58.3–62.3]	43.8 [41.7–45.0]	61.2 [59.9–62.1]	94.1 [90.8–96.1]	77.9 [47.0–79.0]	24.9 [16.8–30.4]	34.4 [33.4–35.4]
Adapters (kNN)	91.6 [91.5–91.7]	68.4 [68.3–68.6]	72.2 [72.1–72.2]	97.7 [97.7–97.8]	91.9 [91.8–92.0]	49.8 [49.8–49.9]	81.1 [78.7–81.8]	83.3 [82.8–83.7]	93.0 [93.0–93.1]	67.3 [58.8–71.8]	81.7 [81.5–81.8]	61.2 [59.1–62.1]	78.0 [75.0–79.5]	44.9 [44.3–46.2]	61.8 [61.3–62.2]	89.3 [87.5–92.8]	78.8 [77.6–79.4]	24.1 [23.7–26.6]	34.5 [30.3–40.7]
Full (EPN)	91.6 [91.2–91.7]	53.2 [53.1–53.4]	63.3 [63.0–63.6]	99.5 [99.5–99.5]	96.1 [96.1–96.2]	55.1 [55.0–55.1]	72.3 [61.7–82.6]	83.3 [82.9–84.1]	94.0 [93.9–94.1]	74.1 [73.9–74.2]	74.5 [74.3–77.1]	56.8 [55.2–57.9]	62.9 [60.6–65.5]	38.2 [37.4–39.5]	64.2 [63.1–64.5]	96.5 [95.5–97.0]	75.5 [74.4–76.1]	22.8 [19.1–23.3]	39.1 [38.7–39.7]
Full (KL)	91.6 [91.5–91.7]	68.4 [68.2–68.6]	64.3 [63.8–64.7]	99.5 [99.5–99.5]	95.9 [95.9–95.9]	55.1 [55.1–55.2]	70.7 [70.2–71.8]	83.1 [81.8–83.5]	93.0 [92.9–93.0]	61.8 [57.4–70.4]	83.4 [83.4–83.5]	54.0 [50.3–55.3]	60.6 [56.5–61.3]	45.1 [41.5–45.2]	64.5 [62.9–65.2]	97.3 [96.8–97.5]	75.0 [74.8–75.4]	24.9 [18.3–25.2]	34.0 [32.0–35.2]
Full (kNN)	91.4 [91.2–91.6]	68.7 [68.4–68.9]	72.2 [72.0–72.3]	99.5 [99.5–99.5]	95.4 [95.3–95.4]	55.1 [55.1–55.2]	75.3 [74.4–77.4]	82.8 [82.5–83.4]	94.8 [94.4–95.1]	73.1 [72.1–73.8]	83.4 [83.4–83.5]	57.3 [55.3–58.1]	54.4 [51.0–57.7]	42.2 [41.8–43.0]	60.6 [58.6–62.2]	93.7 [88.9–95.7]	67.0 [44.0–71.3]	25.5 [25.3–29.0]	41.5 [39.9–42.2]
All Experts (kNN)	91.6 [91.4–91.7]	68.5 [68.4–68.6]	72.1 [72.0–72.3]	99.5 [99.5–99.5]	95.4 [95.3–95.4]	55.1 [55.1–55.2]	77.9 [71.8–80.7]	82.9 [82.6–83.2]	94.9 [94.7–95.0]	73.7 [73.4–73.9]	83.4 [83.3–83.5]	61.0 [58.4–61.8]	77.6 [74.2–80.7]	45.6 [44.8–46.4]	62.3 [60.0–62.9]	87.6 [85.7–90.6]	67.6 [66.8–71.8]	25.3 [25.1–25.6]	41.8 [40.9–42.1]
IN21k																			
Baseline	90.8 [90.7–91.0]	72.5 [72.4–72.6]	71.1 [71.0–71.2]	98.5 [98.4–98.5]	87.6 [87.4–87.7]	48.6 [48.6–48.7]	74.9 [72.6–75.5]	84.3 [84.1–84.5]	87.7 [73.4–94.6]	73.2 [72.9–73.8]	82.8 [82.2–83.0]	52.2 [51.1–53.4]	59.6 [58.1–61.8]	42.1 [37.5–43.2]	61.3 [60.0–62.2]	95.4 [94.4–96.0]	80.5 [78.7–81.6]	30.6 [27.7–30.7]	32.4 [29.7–33.3]
Adapters (kNN)	89.9 [89.7–90.9]	72.4 [72.3–72.6]	71.2 [71.1–71.3]	98.4 [98.4–98.4]	89.4 [89.3–89.6]	49.5 [49.4–49.6]	75.8 [75.3–76.2]	83.9 [83.1–84.3]	94.6 [94.5–94.6]	73.8 [73.2–74.0]	81.8 [80.5–82.0]	54.9 [53.5–55.5]	64.0 [61.1–68.4]	44.9 [44.2–45.2]	61.3 [60.2–61.9]	93.6 [91.4–94.3]	78.6 [77.0–79.5]	27.8 [27.7–31.2]	34.9 [34.4–35.6]
Full (kNN)	90.7 [90.3–90.9]	72.5 [72.5–72.5]	71.4 [71.4–71.5]	98.6 [98.5–98.6]	89.9 [89.4–90.1]	49.7 [49.6–49.7]	74.9 [74.5–77.2]	83.8 [83.5–84.2]	94.9 [94.8–94.9]	73.6 [72.7–74.0]	81.5 [81.5–81.7]	58.2 [55.6–59.4]	68.9 [65.8–70.6]	45.1 [44.5–45.7]	60.9 [59.5–62.0]	95.3 [93.5–96.2]	80.4 [79.7–80.9]	31.1 [30.5–31.8]	35.2 [34.7–35.8]
All Experts (kNN)	90.8 [90.7–90.9]	72.4 [72.2–72.5]	71.4 [71.3–71.4]	98.5 [98.4–98.6]	89.6 [89.4–90.1]	49.5 [49.5–49.6]	76.0 [74.7–77.5]	84.5 [84.0–84.8]	94.8 [94.8–94.9]	73.5 [73.2–74.1]	81.4 [81.3–81.5]	57.3 [55.2–58.5]	68.3 [62.6–72.1]	44.4 [44.3–44.6]	61.4 [60.4–62.0]	93.8 [92.1–95.1]	80.2 [79.6–80.7]	30.6 [30.1–31.3]	34.3 [33.9–35.4]
JFT + IN21k																			
All Experts (kNN)	90.7 [90.6–90.8]	68.5 [68.4–68.7]	71.4 [71.3–71.5]	99.5 [99.5–99.5]	95.4 [95.3–95.4]	55.2 [55.1–55.2]	80.6 [78.1–81.5]	84.5 [84.3–84.8]	94.9 [94.8–94.9]	73.2 [72.1–73.9]	83.5 [83.4–83.6]	61.4 [60.6–62.2]	78.1 [73.2–81.8]	44.8 [44.0–45.4]	62.5 [61.8–63.0]	91.0 [88.1–93.6]	66.9 [66.5–67.9]	30.8 [27.7–31.2]	40.9 [39.4–41.7]

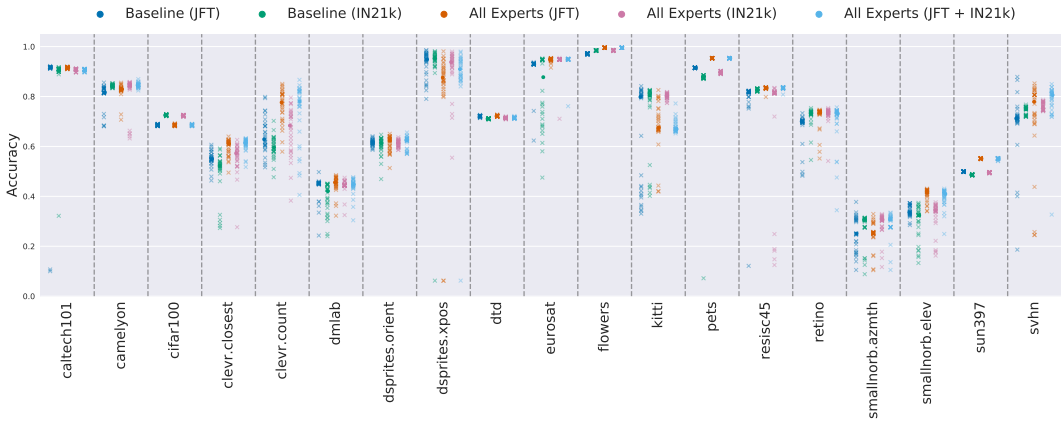


Figure 10: VTAB-1k accuracy in all datasets for 30 runs using different baselines and experts models trained on JFT and ImageNet21k. The median is represented by a darker point. Best seen in color.

F Details on the Comparison with Domain Adaptive Transfer

F.1 Hyperparameters

We randomly explored the space of hyperparameters drawing 36 samples from the following distributions:

- Initial learning rate: Log-uniform in $[2 \cdot 10^{-4}, 2 \cdot 10^{-1}]$.
- Total training steps: Uniform in $\{2000, 4000, 8000, 16000, 32000\}$.
- Weight decay: Log-uniform in $[10^{-6}, 10^{-2}]$.
- Mixup α: Uniform in $\{0, 0.05, 0.1, 0.2, 0.4\}$.

We use a batch size equal to 512 and decay the learning rate by 0.1 at 30%, 60% and 90% of the training duration. During the first 10% of training steps, we linearly warm up the learning rate. Standard data augmentation techniques are applied to prevent overfitting. During training, for all datasets except CIFAR10 (which has a smaller resolution), we resized the images to a fixed size of 512 pixels on both sides, then randomly cropped it to a patch of 480 pixels, randomly flipped the image horizontally, and converted the pixel values to the $[-1, 1]$ range. For CIFAR10, we used a resolution of 160 pixels during the resize and crops of 128 pixels. During evaluation, we simply resize the images and convert the pixel values analogously.

We find the best hyperparameter for each dataset based on the accuracy on the validation data, and then, applying the corresponding set of hyperparameters, the selected expert is fine-tuned again on the union of the training and validation examples of the dataset. The accuracy on a test set is reported. We re-trained the models 30 times using different random seeds.

For downstream training we use a Cloud TPUv3-16. In DAT, the running time depends on the number of steps selected as the hyperparameter and the resolution of the images. At most, it takes 150 minutes to fine-tune one of our experts on one downstream dataset, and at least it takes 8 minutes.

F.2 Detailed results

Table 7 shows the mean accuracy across those 30 trials and the 95% bootstrapped confidence intervals, for each of our experts and selection algorithms, for each dataset as well as the average across the six datasets.

F.3 Differences in asymptotic running time

Table 8 contains an asymptotic analysis of the different phases of each approach. In our case, upstream training includes both the cost of training the generic backbone network for S_U steps, and the cost of

Table 7: Accuracy on the datasets used in Ngiam et al. (2018), and the average accuracy across the six of them. Bootstrapped confidence intervals at 95% are shown below the accuracy, when available. The first two rows are Inception-v3 models, as reported in Ngiam et al. (2018). The rest of the rows are produced by our own models, based on Resnet-50-v2, grouped by expert selection method. The suffixes "2e" and "4e" denote that the expert modules were trained for 2 or 4 epochs, respectively.

	Aircraft	Birds	Cars	CIFAR10	Food	Pets*	Avg.
Baseline	91.4 [91.0–91.7]	78.8 [78.0–79.4]	95.6 [95.4–95.7]	97.8 [97.7–97.9]	91.3 [91.2–91.5]	94.5 [94.4–94.6]	91.6 [91.4–91.7]
kNN							
Adapters, 4e	92.5 [92.2–92.8]	79.4 [78.7–80.1]	95.9 [95.8–96.0]	97.9 [97.8–98.0]	91.6 [91.5–91.7]	94.6 [94.4–94.8]	92.0 [91.9–92.1]
Full, 2e	94.5 [94.2–94.7]	83.5 [83.1–83.9]	96.0 [95.8–96.2]	97.9 [97.8–98.0]	92.9 [92.8–93.1]	96.8 [96.7–96.9]	93.6 [93.5–93.7]
Full, 4e	**94.8** [94.5–95.1]	83.6 [83.1–83.9]	96.1 [96.0–96.3]	97.8 [97.7–97.9]	93.1 [92.8–93.2]	**97.0** [96.9–97.1]	93.7 [93.6–93.8]
KL							
Adapters, 4e	92.1 [91.8–92.5]	80.0 [79.5–80.4]	95.9 [95.8–96.0]	97.9 [97.8–98.0]	91.6 [91.5–91.8]	94.5 [94.3–94.7]	92.0 [91.9–92.1]
Full, 2e	94.6 [93.6–95.0]	83.1 [82.6–83.5]	96.1 [96.0–96.2]	97.9 [97.8–98.1]	92.9 [92.9–93.1]	96.6 [96.5–96.7]	93.5 [93.4–93.7]
Full, 4e	94.4 [94.1–94.7]	83.7 [83.3–84.3]	**96.4** [96.3–96.4]	97.9 [97.8–98.0]	93.1 [92.9–93.3]	96.6 [96.6–96.6]	93.7 [93.6–93.8]
EPN							
Adapters, 4e	92.1 [91.7–92.5]	79.7 [79.1–80.2]	95.8 [95.6–96.0]	97.3 [97.1–97.4]	91.5 [91.3–91.7]	93.1 [91.8–93.6]	91.6 [91.4–91.8]
Full, 2e	94.0 [93.6–94.3]	83.3 [82.7–83.9]	96.2 [96.1–96.2]	96.9 [96.7–97.1]	92.5 [92.3–92.6]	**97.0** [97.0–97.1]	93.3 [93.2–93.4]
Full, 4e	94.2 [94.1–94.4]	84.3 [83.9–84.7]	96.0 [96.0–96.1]	96.6 [96.4–96.7]	92.4 [92.3–92.6]	96.9 [96.9–97.0]	93.4 [93.3–93.5]
Dom-Ad (In-v3) Ngiam et al. (2018)	94.1	81.7	95.7	98.3	94.1	**97.1**	93.5
Dom-Ad (Am-B) Ngiam et al. (2018)	92.8	**85.1**	95.8	**98.6**	**95.3**	96.8	**94.1**

*Pets results are mean per class accuracy as opposed to mean accuracy.

Table 8: Asymptotic running times of Domain Adaptive Transfer (DAT) Ngiam et al. (2018) and our work, where P is the number of parameters of the network, B is the batch size, S_U is the number of training steps of the baseline model, S_A is the number of training steps for adapting the baseline model, S_F is the number of training steps for fine-tuning the specialist model to the downstream task, and E is the number of pre-trained experts in our approach.

	DAT Ngiam et al. (2018)	Ours
Upstream training	$O(S_U \cdot B \cdot P)$	$O((S_U + S_A \cdot E) \cdot B \cdot P)$
Downstream		
Expert preparation	$O((N_T + S_A \cdot B) \cdot P)$	$O((N_T \cdot P + N_T^2) \cdot E)$
Fine-tuning	$O(S_F \cdot B \cdot P)$	$O(S_F \cdot B \cdot P)$

training each of the E experts for S_A steps, with a batch size of B. In the case of Domain Adaptive Transfer (DAT), only the first cost is incurred in this phase. Observe that in our case the cost of upstream training is amortized over the number of tasks that one has to learn, since it's only incurred once.

In the downstream phase, both methods need a forward pass over the number of downstream examples, N_T. Then, leaving the number of parameters of the model aside, the cost of Ngiam et al. (2018) is dominated by $S_A \cdot B$, which is the total number of examples used to fine-tune the baseline model to a weighted/resampled version of the upstream data, and ours is dominated by $N_T \cdot E$, the cost of running a forward pass of the downstream data in each of the pre-trained experts. In practice, because $S_A \cdot B$ (roughly $1.2 \cdot 10^9$, when fine-tuning for 4 epochs on JFT) is much larger than $N_T \cdot E$ (roughly $5 \cdot 10^5$, when using 1 000 downstream examples for selecting over 500 experts), our approach is much faster. Thus, our approach should be roughly three orders of magnitude faster than that of Ngiam et al. (2018), when learning a new task. The final cost of fine-tuning to the downstream dataset is the same in both cases, and it's negligible in comparison.

As described in appendix C.1, we use *a single NVIDIA V100 GPU* as the hardware accelerator when selecting the expert model to use for a given downstream task. It requires less than 2 hours[3] to select a single expert model among a pool of 240 experts, based on 1 000 images from the downstream task. This is the real duration of the "Expert preparation" step, following our method, depicted in table 8.

[3]Most of the time is actually spent reading the model parameters from disk.

Under the same hardware constraints, fine-tuning a R50 on JFT for 4 epochs, using the details described in appendix D, requires around 1780 hours. This is essentially the duration of the "Expert prepartion" step, following the Domain Adaptive Transfer approach[4].

Given that fine-tuning on the downstream dataset takes only a few minutes (see appendix E.2), our transfer approach is $890\times$ faster than that of Domain Adaptive Transfer. Depending on different optimizations and bottlenecks, we expect our approach to be $500\times - 1000\times$ faster in other scenarios.

G THE VALUE OF SEMANTIC EXPERTS

In section 6.4, we have seen that a smart choice of experts leads to substantial gains with respect to a single model trained on all the upstream data. Here, we rule out the possibility that these gains come merely from the fact that we are able to select a representation among a wide range of choices by *directly* testing their initial predictive power on the downstream task. To do so, instead of training our experts in subsets of JFT based on its label hierarchy, we fully fine-tuned the baseline model on 240 uniformly random subsets of JFT, with sizes matching the size of our original semantic experts. We did this independently with both adapter and full experts. Then, we applied kNN to select the best random expert on each downstream dataset. Note this is not at all equivalent to applying transfer at random as in section 6.4.

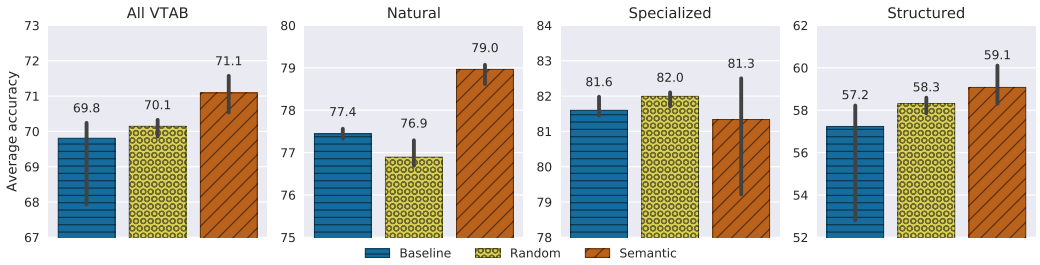
In principle, it was not even clear if this approach with random experts would outperform the baseline. Figure 11 (a) and (b) show our results for adapter and full experts respectively.

In both cases, the overall performance drops when we replace semantic experts with random ones. This difference seems stronger in the case of adapter-based experts. However, notice that the full expert results are very influenced by strong negative results in one of the structured datasets (Clevr Count), as shown in table 6. Also, random experts results are comparable to the baseline. Note that random experts are not dumb; they are just a diverse set of models with the general flavor of the upstream dataset. The algorithm can still benefit from their diversity when confronted with a new task, whereas we expect them to be more similar to each other than in the semantic case.

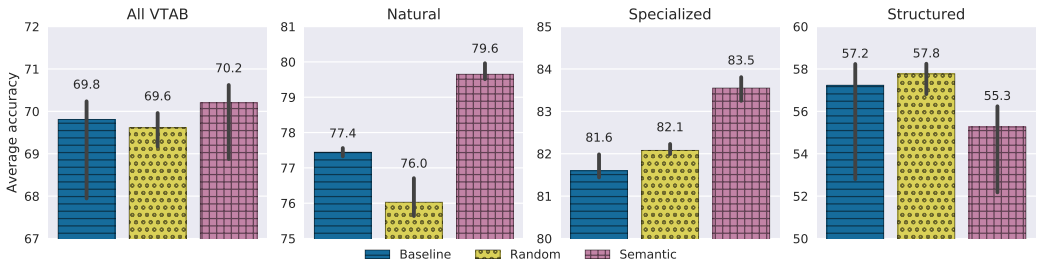
The semantic experts are trained mostly on natural slices, and we see a large improvement in the NATURAL tasks when we use them (2.7% and 4.7% gains for adapters and full, respectively, compared to random experts). This reinforces the idea that experts in the right domain can be very helpful. Moreover, in NATURAL tasks, the baseline outperforms random experts; this suggests that here more data is better unless data is smartly selected.

As we have hypothesized before, it seems that our natural-image experts do not provide a meaningful expertise or competitive edge on STRUCTURED tasks. We see a large improvement on SPECIALIZED tasks when using full experts, while the effect is not there for adapters. Accordingly, we would not read too much into these results.

[4]In fact, we are ignoring the cost of the forward pass through the 1 000 downstream datasets, but that cost is negligible compared to fine-tuning on JFT.



(a) JFT Adapter Experts



(b) JFT Full Experts

Figure 11: Results on VTAB with 1 000 examples per dataset achieved by experts trained on random subsets of JFT and experts trained on semantically meaningful subsets. For each dataset in VTAB, the median accuracy over 30 trials is considered. The results of the datasets in each group are averaged. The error bars show the (percentile) bootstrapped confidence intervals at 95% level.