Supplemental Material for paper ID 628
A Flexible Selection Scheme for Minimum-Effort Transfer Learning

This supplemental material contains complementary experiments for the paper "Flex-Tuning: A Flexible and Automatic Selection Scheme for Transfer Learning". The first section contains the detailed validation accuracies for each of the N_{prox} models which are built during the model selection phase of the fast and faster flextuning variants. In the following sections, we report detailed results for each of our experimental settings; This includes the dataset construction, experimental settings, and quantitative and qualitative results, as well as for the retrieval experiments. We first detail our experiments on a small illustrative 4-layers network on MNIST [9] in Section 2. In Section 3, we report on experiments for a middle size 7-layers network, using the CIFAR-10 [8] and Quick, Draw! [1] datasets. Finally, we describe large-scale experiments with the Inception2 architecture on the ILSVRC [13] and PACS [10] datasets in Section 4 and Section 5, respectively.

In Section 6, we report complete results of our information retrieval experiments. Finally, the last section contains results for the transfer learning baseline described in [2]. In our experiments, it did not perform much better than the standard finetuning baseline therefore we did not include it in the main manuscript.

1. Model selection in fast and faster flextuning

As described in the main manuscript, the proposed *fast flextuning* variant selects the best unit to tune by the following "network surgery": The method starts by training one new model, $N_{\texttt{ft-all}}$, by fine-tuning all units of the pre-trained network on the training data available for the target domain (In the *faster* variant, this model is only trained for one epoch). From this, we construct L new networks: for any $l = 1, \ldots, L$, we create a proxy network, $N_{\texttt{prox-}l}$, by copying all units from N, except the l-th one, which is copied from the fine-tuned network, $N_{\texttt{ft-all}}$.

The construction allows us to derive a measure which of the network units is the most promising candidate for fine-tuning, namely the one that leads to the *biggest improvement in accuracy (if any) when applied to the target domain*. Numerically, we compute the accuracy of each model $N_{\texttt{prox-}l}$ on the validation dataset and identify the value for l with highest accuracy. We report validation accuracies of these models in Figure 1 to Figure 4.

Each of this plot represents the validation accuracy versus layer l obtained by model $N_{\texttt{prox-}l}$. Each line corresponds to a different source→domain shift setting. The unit selected by the method is highlighter in white. Finally, the point at $l = 0$ corresonds to applying the source pre-trained network directly to the target samples.

In general, we note that the selected unit is rather consistent across data subsamling ratios (left to right plot). However, the different domain shifts (different lines) vary quite a lot. For some settings, for instance YUV or HSV ILSVRC, the influence of network surgery is clear and significantly highlight a particular unit. While for more complex scenarios such as *photo→art*, several units have strong influence. This phenomenon is even more present in the faster flextuning plots.

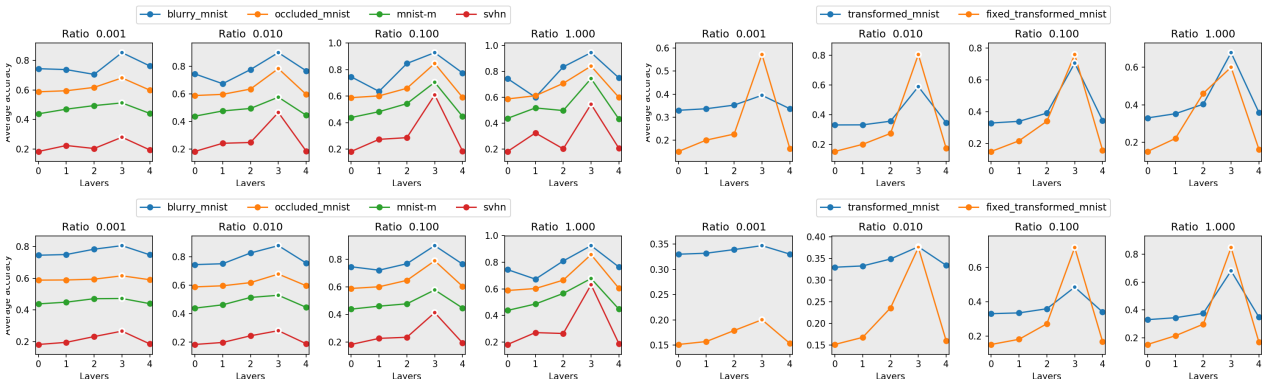


Figure 1: $N_{\texttt{prox-}l}$ validation accuracies for the fast (top) and faster (bottom) flextuning model selection criterion for MNIST

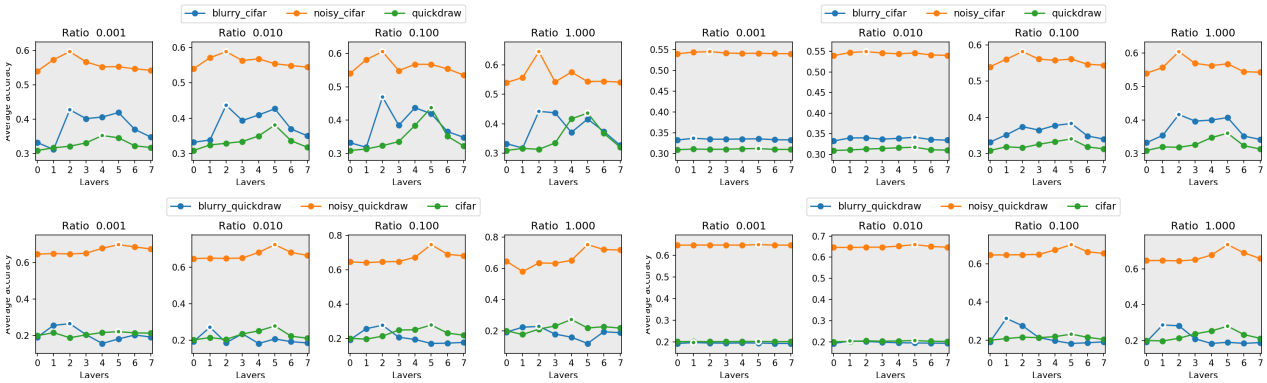


Figure 2: $N_{\text{prox-}l}$ validation accuracies for the fast (left) and faster (right) flextuning model selection criterion for CIFAR and Quickdraw

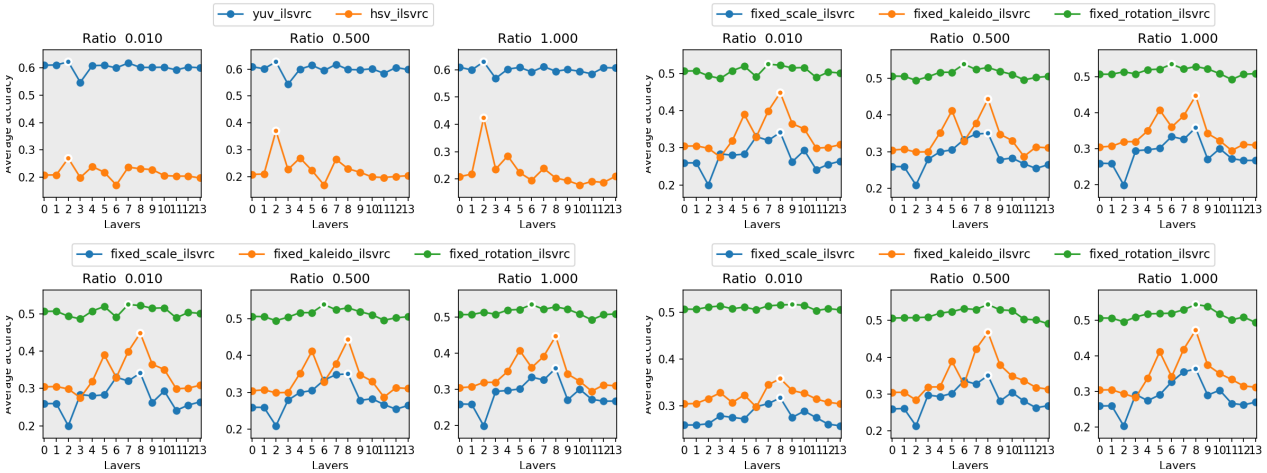


Figure 3: $N_{\text{prox-}l}$ validation accuracies for the fast (top) and faster (bottom) flextuning model selection criterion for ILSVRC

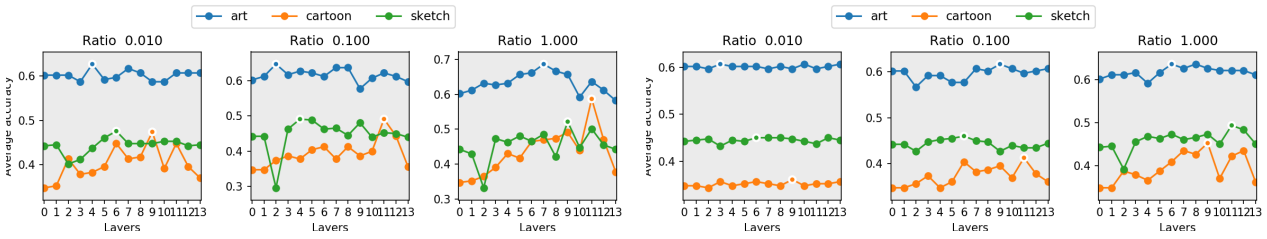


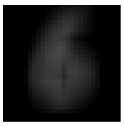
Figure 4: $N_{\text{prox-}l}$ validation accuracies for the fast (lefgt) and faster (faster) flextuning model selection criterion for PACS

2. MNIST experiments

Datasets and base architecture. We build a network composed of 2 convolutional layers followed by 2 fully connected layers, with ReLU activations. We pretrain this network on a random subset of 25000 images from the original MNIST training set, achieving **0.989** top-1 accuracy on the test set. We split the remaining images in a training set of 30000 images, $\mathcal{D}_{\text{train}}$, and a validation set of 5000 images, $\mathcal{D}_{\text{val}}$, which we use to generate synthetic transforms of MNIST as target domains, avoiding any overlap with the dataset used for pretraining. We denote by $\mathcal{D}_{\text{test}}$ the original MNIST test set of 10000 images.

We first build **Blurry MNIST** by applying a Gaussian blur with a kernel of length 8 and standard deviation $\sigma = 1$ on all images in $\mathcal{D} = (\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}}, \mathcal{D}_{\text{test}})$. We also construct **Occluded MNIST** by randomly occluding a patch of size 14x14 pixels on each image in $\mathcal{D}$. The random patches coordinates are generated only once hence the dataset stays the same across runs. We also generate two **geometric transforms** of MNIST by creating affine transformations, either one fixed transformation for all images or a different random transformation for each image, and applying them to $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{val}}$ and $\mathcal{D}_{\text{test}}$. These affine transforms are generated as a rescaling operation ($s \in [0.4, 1.0]$) followed by a translation, randomly sampled from any value that keeps the sampled patch in the original 28x28 pixels canvas, and finally a rotation, with angle $\theta \in [-1.5, 1.5]$ radians. As for more complex transformations, we follow the same construct as for the **MNIST-M** dataset [5] by inlaying random background patches from the BSD [11] dataset on $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{val}}$ and $\mathcal{D}_{\text{test}}$. We also experiment on **SVHN** [12] as a target domain, which we split into 63k training images, 10k validation images, and keep the original test set of 26k images. In this setting, we also resize images to size 28x28 pixels to match the other datasets. In order to evaluate the impact of data scarcity, we create subsampled versions of each target training dataset by randomly sampling images with ratios 0.001, 0.01 and 0.1.

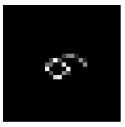
Source domain
MNIST (subset)
25k images, 10 classes
4-layers network
top-1: 0.989



Blurry
top-1: 0.748
$\cos_{\text{src}}$: 0.101
$\cos_{\text{pc}}$: 0.116
ipc: 3-30-300-3k

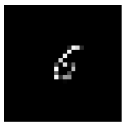
Occluded
top-1: 0.581
$\cos_{\text{src}}$: 0.051
$\cos_{\text{pc}}$: 0.067
ipc: 3-30-300-3k

MNIST-M [4]
top-1: 0.439
$\cos_{\text{src}}$: 0.128
$\cos_{\text{pc}}$: 0.210
ipc: 3-30-300-3k



Random transform
top-1: 0.322
$\cos_{\text{src}}$: 0.283
$\cos_{\text{pc}}$: 0.375
ipc: 3-30-300-3k

SVHN [12]
top-1: 0.211
$\cos_{\text{src}}$: 0.248
$\cos_{\text{pc}}$: 0.435
ipc: 6-60-600-6k



Fixed transform
top-1: 0.160
$\cos_{\text{src}}$: 0.443
$\cos_{\text{pc}}$: 0.531
ipc: 3-30-300-3k

Table 1: Experimental setup for the MNIST source domain (left). For each target domain (right), we report the average number of training images per class (ipc) for each data subsampling ratio. As a quantitative measure of domain shift, we order target domains in decreasing order of their test accuracy under the pretrained source network, which generally corresponds well to human intuition of each task difficulty. We also draw inspiration from [3], which proposes to measure domain shift as the cosine distance between the mean responses in the before-to-last fully-connected layer of the source network applied to the source and target domain, denoted by $\cos_{\text{src}}$. We also report its per-class variant, $\cos_{\text{pc}}$.

Results. Quantitative results for all subsampling ratios and target domains are reported in Table 2, comparing flex-tuning and its variants to the fine-tuning baselines. As explained in the main paper, flex and fast-flex usually choose to fine-tune all layers (*i.e.* they recover ft-all), which achieves best accuracy out of the network-tuning method. While this is almost always the case in the full dataset scenarios, it happens less often in the small sample size settings, where there is a higher risk of overfitting when fine-tuning the whole network. Secondly, the overall best performing method is prep-tuning. In fact, the pre-processing module easily recovers from some domain shifts such as Blurry MNIST or Fixed Transformed MNIST, significantly improving the accuracies in these settings, especially in the small sample size scenarios.

In Table 3 and Figure 5, we report quantitative and qualitative results for the pre-processing module alone, without flex-tuning model selection. For the geometric transforms, we define this module as a Spatial Transformer Network [7] with a 1-layer encoder. Given an input image, the encoder outputs parameters for an affine transformation of the same form as the ones we considered for generating the shifted domains (*i.e.* a rescaling operations followed by a translation followed by a rotation). For the other, more generic, transformations, we consider a pix2pix architecture [6] taking inputs of size 32x32 pixels, with n layers in the encoder and n layers in the decoder (n ranging from 1 to 3). As mentioned in the original pix2pix paper, we tried using both upscaling + convolution and transpose convolutions. The latter usually yields slightly better classification accuracies but not by a significant margin and generates a lot of artifacts, hence we use the first method, which indeed generates more visually pleasing images. Finally we use standard batch normalization instead of instance normalization, as it did not significantly impact nor classification accuracies nor generated images in our experiments. As can be seen in these figures, when enough data is available, making use of all parameters in flex-prep (3) performs best. For smaller sample sizes, training a reduced number of parameters such as in flex-prep (2) is more advantageous.

MNIST		flex				ft-			
		flex	fast	faster	prep	fc (1)	fc (2)	ss	all
Blurry	(0.75)	0.890 ± 0.00	0.860 ± 0.00	0.857 ± 0.01	**0.950 ± 0.00**	0.867 ± 0.00	0.846 ± 0.01	0.862 ± 0.00	0.851 ± 0.00
Occluded	(0.58)	**0.695 ± 0.00**	0.664 ± 0.01	0.661 ± 0.01	**0.695 ± 0.00**	0.644 ± 0.01	0.654 ± 0.01	0.662 ± 0.00	0.660 ± 0.01
MNIST-M	(0.44)	0.564 ± 0.00	0.564 ± 0.00	0.528 ± 0.00	**0.643 ± 0.00**	0.524 ± 0.00	0.523 ± 0.01	0.540 ± 0.01	0.528 ± 0.00
SVHN	(0.21)	**0.426 ± 0.00**	**0.426 ± 0.00**	0.414 ± 0.00	**0.426 ± 0.00**	0.365 ± 0.01	0.412 ± 0.00	0.403 ± 0.00	**0.426 ± 0.00**
Transform (rnd)	(0.32)	0.400 ± 0.00	0.400 ± 0.00	0.399 ± 0.00	**0.498 ± 0.00**	0.391 ± 0.01	0.401 ± 0.00	0.395 ± 0.01	0.396 ± 0.00
Transform (fixed)	(0.16)	0.776 ± 0.00	0.776 ± 0.00	0.770 ± 0.00	**0.901 ± 0.00**	0.743 ± 0.00	0.768 ± 0.00	0.752 ± 0.00	0.776 ± 0.00

(a) Data subsampling ratio 0.001 ($\sim$ 3 images per class). Metrics are reported for 20 repetitions of the experiment with different sampled training images.

MNIST		flex				ft-			
		flex	fast	faster	prep	fc (1)	fc (2)	ss	all
Blurry	(0.75)	0.926	0.926	0.926	**0.957**	0.921	0.928	0.928	0.921
Occluded	(0.58)	**0.806**	**0.806**	0.801	**0.806**	0.785	0.801	0.792	**0.806**
MNIST-M	(0.44)	0.683	0.683	0.671	**0.776**	0.615	0.670	0.675	0.683
SVHN	(0.21)	**0.669**	**0.669**	0.572	**0.669**	0.451	0.595	0.657	**0.669**
Transform (rnd)	(0.32)	0.644	0.644	0.644	**0.666**	0.573	0.638	0.634	0.625
Transform (fixed)	(0.16)	0.908	0.887	0.879	**0.937**	0.839	0.875	0.866	0.887

(b) Data subsampling ratio 0.01 ($\sim$ 30 images per class).

MNIST		flex				ft-			
		flex	fast	faster	prep	fc (1)	fc (2)	ss	all
Blurry	(0.75)	0.967	0.967	0.967	**0.981**	0.941	0.964	0.957	0.965
Occluded	(0.58)	0.873	0.873	0.873	0.873	0.849	**0.875**	0.865	0.870
MNIST-M	(0.44)	0.828	0.828	0.795	**0.867**	0.704	0.797	0.783	0.828
SVHN	(0.21)	**0.849**	**0.849**	0.812	**0.849**	0.558	0.800	0.798	**0.849**
Transform (rnd)	(0.32)	**0.843**	**0.843**	0.830	**0.843**	0.683	0.836	0.760	**0.843**
Transform (fixed)	(0.16)	0.955	0.955	0.947	**0.973**	0.876	0.945	0.916	0.955

(c) Data subsampling ratio 0.1 ($\sim$ 300 images per class).

MNIST		flex				ft-			
		flex	fast	faster	prep	fc (1)	fc (2)	ss	all
Blurry	(0.75)	0.987	0.987	0.981	**0.988**	0.955	0.982	0.970	0.987
Occluded	(0.58)	**0.917**	**0.917**	0.915	**0.917**	0.867	0.912	0.893	**0.917**
MNIST-M	(0.44)	0.895	0.895	0.857	**0.953**	0.739	0.868	0.846	0.895
SVHN	(0.21)	**0.909**	**0.909**	0.851	**0.909**	0.670	0.877	0.848	**0.909**
Transform (rnd)	(0.32)	**0.941**	**0.941**	0.900	**0.941**	0.733	0.909	0.835	**0.941**
Transform (fixed)	(0.16)	0.979	0.979	0.963	**0.980**	0.900	0.969	0.943	0.979

(d) Experiments using the full target domain training dataset ($\sim$ 3000 images per class).

Table 2: Quantitative results for the MNIST source domain. We report on experiments with the proposed *flex-tuning* (`flex`), its two faster variants *fast flex-tuning* (`fast-flex`) and *even faster flex-tuning* (`faster-flex`), as well as flex-tuning augmented with a *preprocessing module* (`flex-prep`). We compare against transfer learning baselines: fine-tuning *all layers* in the architecture (`ft-all`), versus fine-tuning only *fully-connected* ones, either only the last one (`ft-fc` (1)) or both of them (`ft-fc` (2)), and finally adding scale and shift operations (`ft-ss`).

MNIST	Ratio 0.001			Ratio 0.010			Ratio 0.100			Ratio 1.000		
	prep (3)	prep (2)	prep (1)	prep (3)	prep (2)	prep (1)	prep (3)	prep (2)	prep (1)	prep (3)	prep (2)	prep (1)
Blurry	0.817	**0.950**	0.917	0.939	**0.957**	0.955	0.974	**0.981**	0.967	0.987	**0.988**	0.975
Occluded	0.452	**0.536**	0.504	0.686	**0.703**	0.646	0.828	**0.830**	0.729	**0.903**	0.897	0.791
MNIST-M	0.410	0.630	**0.643**	0.638	**0.776**	0.725	0.840	**0.867**	0.843	**0.953**	0.949	0.865
SVHN	0.227	**0.261**	0.244	**0.476**	0.418	0.369	**0.752**	0.664	0.431	**0.881**	0.833	0.469
Transform (rnd)	**0.498**	-	-	**0.666**	-	-	**0.826**	-	-	**0.924**	-	-
Transform (fixed)	**0.901**	-	-	**0.937**	-	-	**0.973**	-	-	**0.980**	-	-

Table 3: Quantitative results from the pre-processing module experiments for different architectures and subsampling ratio For the pix2pix network, `prep` (n) designates an architecture with n layers in the encoder and n layers in the decoder.

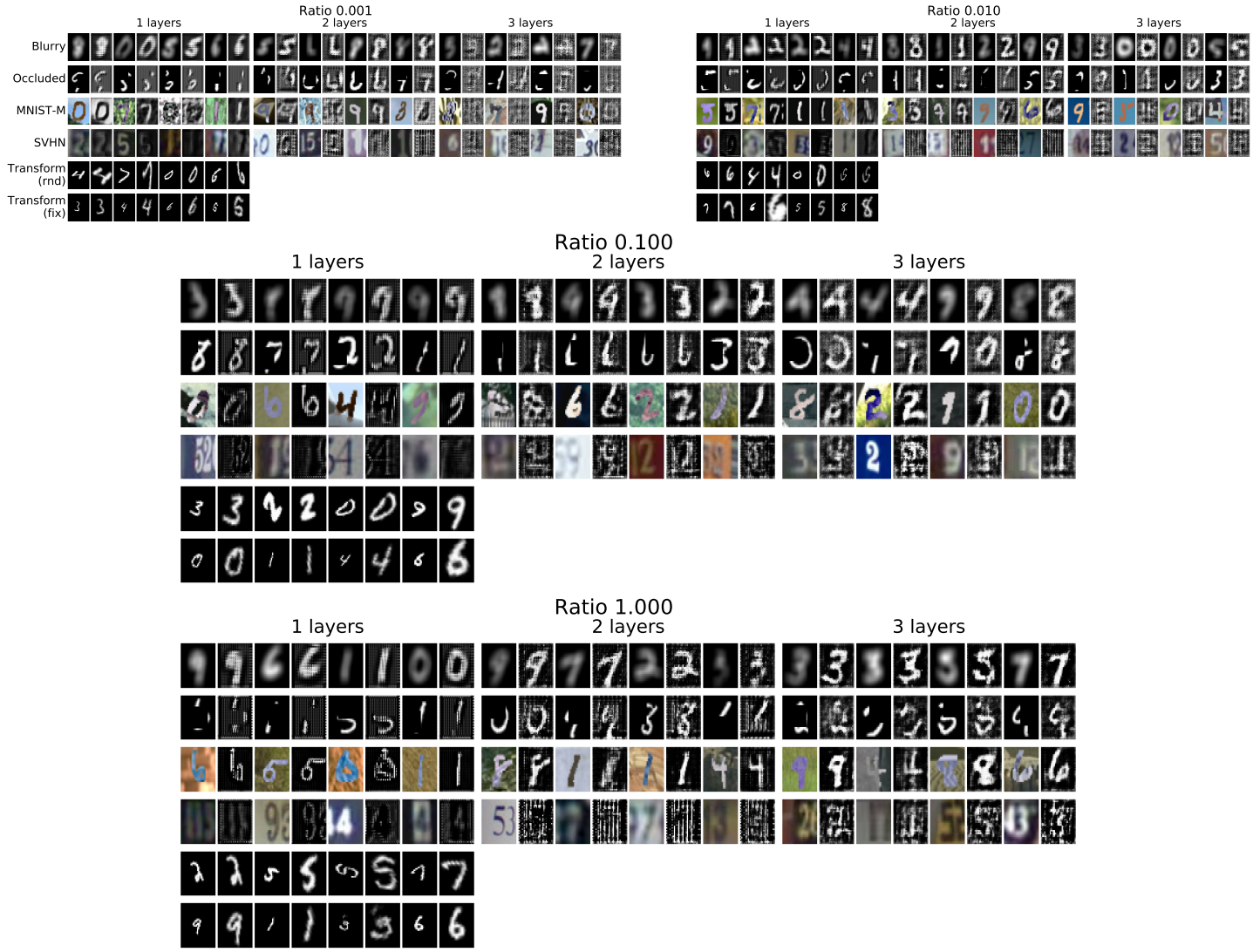


Figure 5: Qualitative results obtained with the pre-processing module for each subsampling ratio, target domain and pre-processing module architecture. Figures reads row-wise, two-by-two, where each pair contains the input image on the left (target domain) and the generated pre-processed image on the right (source domain). Best seen in PDF with zoom.

3. CIFAR and Quick, Draw! experiments

Datasets and base architecture. We experiment on a standard middle-sized architecture composed of 5 convolutional layers followed by 2 fully-connected ones. Each layer is equipped with ReLU activations, except for the last one, and each convolutional layer is followed by max-pooling. We consider as source domains the CIFAR-10 [8] and Quick, Draw! [1] datasets. As we will use each of them as target domain for the other, we restrict ourselves to classes they have in common, *i.e*. all CIFAR classes except one: "airplane", "automobile", "bird", "cat", "dog", "frog", "horse", "ship" and "truck".

For **CIFAR**, we first randomly split the training set in two. We use the first split, containing 17991 images, to pre-train the network. The final model reaches **0.738** top-1 accuracy on the test set. We further divide the remaining split in two: A training set of 18006 images, $\mathcal{D}_{\text{train}}$, which we use to generate synthetic domain shifts, and a validation set, $\mathcal{D}_{\text{val}}$, of size 9003. We use the original test split of CIFAR containing 9000 images for testing purposes.

For the **Quick, Draw!** dataset, we first extract 10000 random images from each of the nine categories of interest and resize them to size 32x32 pixels: using the full original dataset of 50 million images would create a huge discrepancy in terms of dataset size. We again split this set of images into 40000 image to pre-train the source model on Quick, Draw!, reaching **0.654** top-1 test accuracy, a set of 30000 images, $\mathcal{D}_{\text{train}}$, that we use for creating synthetic target domain training sets, a set of 5000 images used for validation purposes, $\mathcal{D}_{\text{val}}$, and finally the remaining 15000 images we use as our test split, $\mathcal{D}_{\text{test}}$.

Starting from CIFAR as a source domain, we then create the following target domains: To obtain **Blurry CIFAR**, we apply a Gaussian blur with a kernel of length 4 and standard deviation $\sigma = 0.5$ on all images in $\mathcal{D} = (\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}}, \mathcal{D}_{\text{test}})$. Similarly, we create **Noisy CIFAR** by adding random-generated Gaussian noise with mean 0 and standard deviation 30 (images ranging in [0, 255]) to each image in $\mathcal{D}$. Note that the transformation is done once so that the dataset is the same for all settings. Finally, we create a **Quick, Draw!** target domain by simply using the $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{val}}$ and $\mathcal{D}_{\text{test}}$ for the Quick, Draw! dataset as defined in the previous paragraph. For the Quick, Draw! dataset as the source domain, we define target domains **Blurry QuickDraw**, **Noisy QuickDraw** and **CIFAR** in the same manner.

As for the MNIST experiments, we also create subsampled versions of each target training dataset, $\mathcal{D}_{\text{train}}$, by randomly sampling images with ratios 0.001, 0.01 and 0.1. A summary of our experimental setup is given in Table 4.

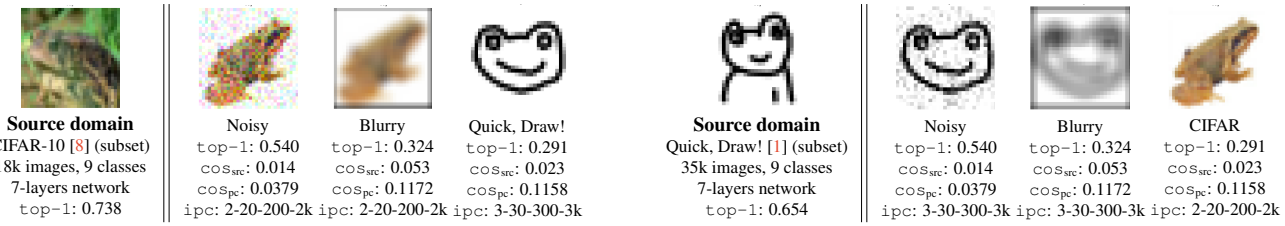


Table 4: CIFAR (left) and Quick, Draw! (right) based domain shifts we consider in our experimental setup.

Results. Quantitative results for all subsampling ratios and target domains are reported in Table 5 for experiments using CIFAR as their source domain, and in Table 6 for experiments with Quick, Draw! as the source domain. As in Section 2, we observe that in the full dataset scenario, fine-tuning all layers yields best performance, and is indeed chosen by flex-tuning and fast flex-tuning. As the sample size decreases, this is however not always true and a few settings benefit from fine-tuning a specific unit in the network instead.

In Table 7, Figure 7 and Figure 6, we report quantitative and qualitative results for the pre-processing module experiment. We use the same Pix2Pix network as described in Section 2. We observe that the pre-processing module performs well for the scenarios with Quick, Draw! as their source domain, most likely due to the simple appearance of this domain. In the other, more complex, settings, it fails to nicely recover the source CIFAR domain, even when the full dataset is available, which leads to poor classification accuracies.

CIFAR	flex				ft-			
	flex	fast	faster	prep	fc (1)	fc (2)	ss	all
Blurry (0.32)	**0.514 ± 0.00**	**0.514 ± 0.00**	0.344 ± 0.02	**0.514 ± 0.00**	0.408 ± 0.00	0.427 ± 0.00	0.490 ± 0.00	0.476 ± 0.01
Noisy (0.54)	**0.618 ± 0.00**	0.618 ± 0.00	0.618 ± 0.00	**0.618 ± 0.00**	0.555 ± 0.00	0.566 ± 0.00	0.603 ± 0.01	0.582 ± 0.00
QuickDraw (0.29)	**0.392 ± 0.00**	0.391 ± 0.00	0.386 ± 0.00	**0.392 ± 0.00**	0.359 ± 0.00	0.380 ± 0.01	0.378 ± 0.01	0.391 ± 0.00

(a) Data subsampling ratio 0.001 ($\sim$ 2 images per class). Metrics are reported for 20 repetitions of the experiment with different sampled training images.

CIFAR	flex				ft-			
	flex	fast	faster	prep	fc (1)	fc (2)	ss	all
Blurry (0.32)	**0.577**	**0.577**	0.512	**0.577**	0.444	0.501	0.569	**0.577**
Noisy (0.54)	**0.624**	**0.624**	**0.624**	**0.624**	0.583	0.597	0.618	0.621
QuickDraw (0.29)	0.518	0.517	0.517	0.518	0.475	**0.525**	0.495	0.501

(b) Data subsampling ratio 0.01 ($\sim$ 20 images per class).

CIFAR	flex				ft-			
	flex	fast	faster	prep	fc (1)	fc (2)	ss	all
Blurry (0.32)	0.609	0.608	0.566	0.609	0.525	0.552	**0.623**	0.608
Noisy (0.54)	0.644	0.629	0.629	0.644	0.602	0.620	**0.653**	0.613
QuickDraw (0.29)	0.663	0.667	0.667	0.663	0.569	0.662	0.643	**0.671**

(c) Data subsampling ratio 0.01 ($\sim$ 200 images per class).

CIFAR	flex				ft-			
	flex	fast	faster	prep	fc (1)	fc (2)	ss	all
Blurry (0.32)	**0.689**	**0.689**	0.644	**0.689**	0.560	0.609	0.685	**0.689**
Noisy (0.54)	0.685	0.685	0.651	0.685	0.633	0.640	**0.705**	0.685
QuickDraw (0.29)	**0.786**	**0.786**	0.744	**0.786**	0.581	0.721	0.702	**0.786**

(d) Experiments using the full target domain training dataset ($\sim$ 3000 images per class).

Table 5: Quantitative results for the CIFAR source domain. We report on experiments with the proposed *flex-tuning* (`flex`), its two faster variants *fast flex-tuning* (`fast-flex`) and *even faster flex-tuning* (`faster-flex`), as well as flex-tuning augmented with a *preprocessing module* (`flex-prep`). We compare against transfer learning baselines: fine-tuning *all layers* in the architecture (`ft-all`), versus fine-tuning only *fully-connected* ones, either only the last one (`ft-fc` (1)) or both of them (`ft-fc` (2)).

QuickDraw	flex				ft-			
	flex	fast	faster	prep	fc (1)	fc (2)	ss	all
Blurry (0.19)	**0.560 ± 0.01**	**0.560 ± 0.01**	0.525 ± 0.00	**0.560 ± 0.01**	0.290 ± 0.00	0.325 ± 0.00	0.327 ± 0.00	0.386 ± 0.00
Noisy (0.63)	0.763 ± 0.00	0.760 ± 0.00	0.758 ± 0.00	0.763 ± 0.00	0.766 ± 0.00	0.768 ± 0.00	**0.782 ± 0.01**	0.757 ± 0.01
CIFAR (0.20)	0.241 ± 0.00	0.241 ± 0.00	0.230 ± 0.00	0.241 ± 0.00	0.203 ± 0.00	0.228 ± 0.00	**0.264 ± 0.01**	0.241 ± 0.00

(a) Data subsampling ratio 0.001 ($\sim$ 2 images per class). Metrics are reported for 20 repetitions of the experiment with different sampled training images.

QuickDraw	flex				ft-			
	flex	fast	faster	prep	fc (1)	fc (2)	ss	all
Blurry (0.19)	0.642	0.631	0.560	**0.642**	0.426	0.468	**0.707**	0.631
Noisy (0.63)	0.801	0.801	0.795	0.801	0.788	0.792	**0.805**	0.801
CIFAR (0.20)	**0.424**	**0.424**	0.401	**0.424**	0.333	0.347	0.388	**0.424**

(b) Data subsampling ratio 0.01 ($\sim$ 20 images per class).

QuickDraw	flex				ft-			
	flex	fast	faster	prep	fc (1)	fc (2)	ss	all
Blurry (0.19)	**0.726**	0.722	0.595	**0.726**	0.471	0.583	0.724	0.722
Noisy (0.63)	**0.815**	0.797	0.797	**0.815**	**0.815**	0.797	0.812	0.772
CIFAR (0.20)	**0.601**	**0.601**	0.532	**0.601**	0.394	0.413	0.543	**0.601**

(c) Data subsampling ratio 0.01 ($\sim$ 200 images per class).

QuickDraw	flex				ft-			
	flex	fast	faster	prep	fc (1)	fc (2)	ss	all
Blurry (0.19)	**0.776**	**0.776**	0.613	**0.776**	0.453	0.629	0.658	**0.776**
Noisy (0.63)	0.817	**0.822**	**0.822**	0.817	0.789	0.818	0.814	0.794
CIFAR (0.20)	**0.718**	**0.718**	0.632	**0.718**	0.432	0.529	0.697	**0.718**

(d) Experiments using the full target domain training dataset ($\sim$ 3000 images per class).

Table 6: Quantitative results for the Quick, Draw! source domain. We report on experiments with the proposed *flex-tuning* (`flex`), its two faster variants *fast flex-tuning* (`fast-flex`) and *even faster flex-tuning* (`faster-flex`), as well as flex-tuning augmented with a *preprocessing module* (`flex-prep`). We compare against transfer learning baselines: fine-tuning *all layers* in the architecture (`ft-all`), versus fine-tuning only *fully-connected* ones, either only the last one (`ft-fc` (1)) or both of them (`ft-fc` (2)).

CIFAR	Ratio 0.001			Ratio 0.010			Ratio 0.100			Ratio 1.000		
	prep (3)	prep (2)	prep (1)	prep (3)	prep (2)	prep (1)	prep (3)	prep (2)	prep (1)	prep (3)	prep (2)	prep (1)
Blurry	0.123	0.165	**0.184**	0.147	**0.283**	0.246	**0.433**	0.342	0.341	**0.598**	0.568	0.382
Noisy	0.118	0.164	**0.176**	**0.228**	0.206	0.210	0.408	0.279	**0.463**	**0.552**	0.498	0.534
QuickDraw	0.227	0.216	**0.256**	**0.421**	0.400	0.264	**0.596**	0.548	0.435	**0.764**	0.722	0.449

QuickDraw	Ratio 0.001			Ratio 0.010			Ratio 0.100			Ratio 1.000		
	prep (3)	prep (2)	prep (1)	prep (3)	prep (2)	prep (1)	prep (3)	prep (2)	prep (1)	prep (3)	prep (2)	prep (1)
Blurry	0.285	0.338	**0.523**	0.458	0.550	**0.571**	0.633	**0.639**	0.618	**0.722**	0.691	0.646
Noisy	0.330	0.415	**0.537**	0.579	0.537	**0.585**	**0.651**	0.622	0.635	**0.739**	0.714	0.652
CIFAR	0.140	0.125	**0.167**	**0.219**	0.175	0.207	**0.410**	0.309	0.265	**0.622**	0.437	0.305

Table 7: Quantitative results for the pre-processing module experiments on CIFAR and Quick, Draw! source domains

Figure 6: Qualitative results from the pre-processing module experiments with CIFAR as the source domain.

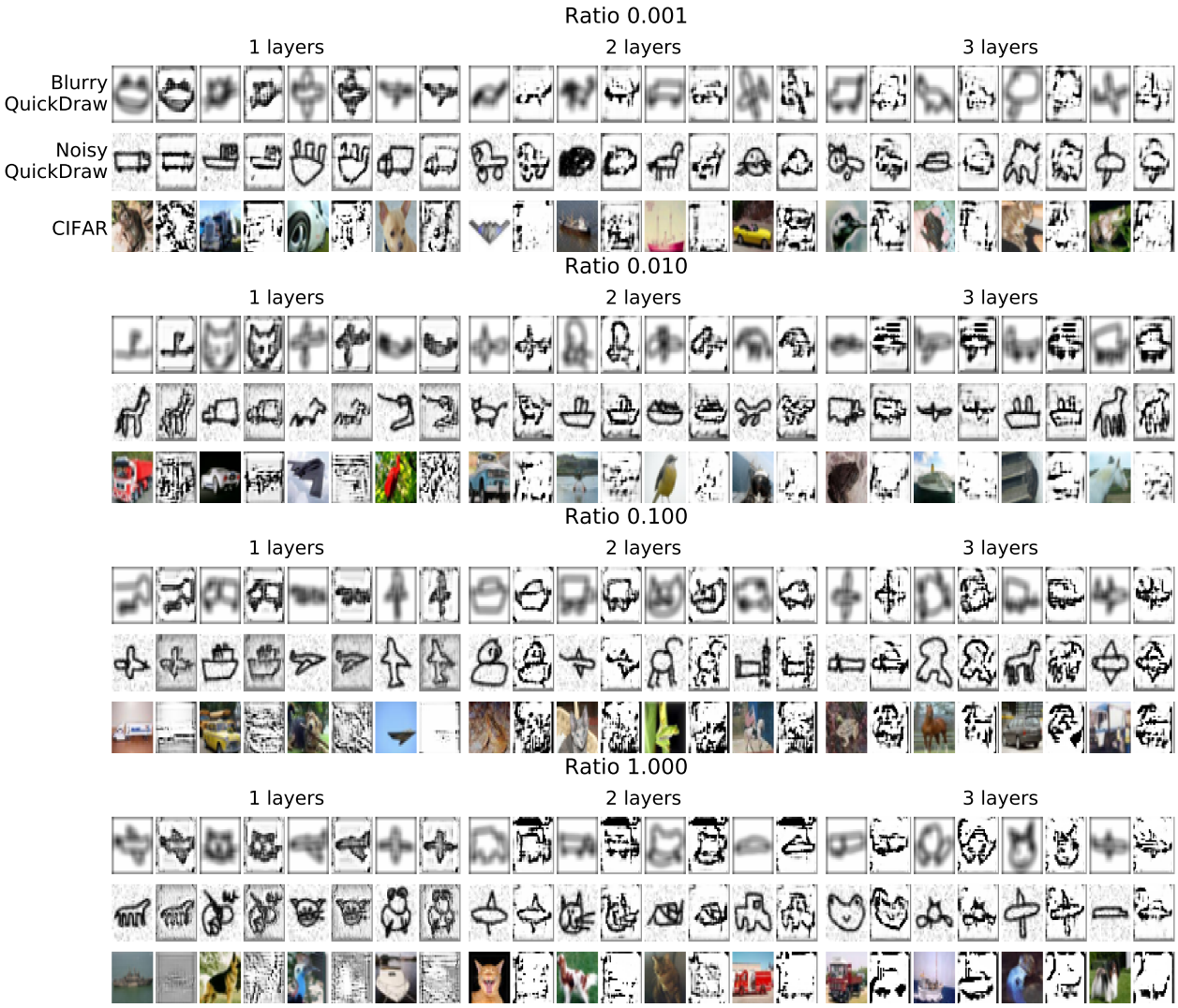


Figure 7: Qualitative results from the pre-processing module experiments with Quick, Draw! as the source domain.

4. ILSVRC experiments

Datasets and base architecture. For this setting, we employ a standard Inception2 architecture, pre-trained on the ILSVRC12 train split, and implemented in the tensornets framework [14]. This source pre-trained network reaches **0.918** top-5 accuracy on the test set. For all experimental settings, we also follow the same pre-processing as used for the original network: we resize all images to 256x256 pixels and feed the network with central crop of size 224x224px.

To generate target domains, we first split the ILSVRC12 validation split into three random sets: 25k images for training, $\mathcal{D}_{\text{train}}$, 5k images for validation, $\mathcal{D}_{\text{val}}$, and finally 20k images for testing purposes, $\mathcal{D}_{\text{test}}$. We first generate **YUV ILSVRC** and **HSV ILSVRC** by converting all images from $\mathcal{D} = (\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}}, \mathcal{D}_{\text{test}})$ from RGB to YUV and HSV color spaces, respectively. We then generate three geometric transformed target domains: **Fixed Scaling (stretch)** is obtained by rescaling all images with the same fixed coefficient $s = 0.5$ and padding the empty pixels by repeating the edges values of the images. **Fixed Scaling (symmetric)** is obtained similarly but using the SYMMETRIC padding mode of TensorFlow. Finally, **Fixed Rotation** corresponds to applying the same fixed rotation with angle 0.6 radians to all images.

We also create subsampled versions of each target training dataset by randomly sampling images with ratios 0.04 ($\sim$ 1 image per class) and 0.5 ($\sim$ 12 images per class). A summary of our experimental setup is given in Table 8.

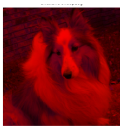
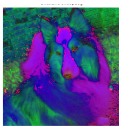
					
Source domain **ILSVRC** [13] **('12 train split)** 1M images, 1000 classes Inception2 top-5: 0.918	YUV top-5: 0.841 $\cos_{\text{src}}$: 0.0231 $\cos_{\text{pc}}$: 0.079 ipc: 2 - 12 - 25	Fixed rotation top-5: 0.743 $\cos_{\text{src}}$: 0.0104 $\cos_{\text{pc}}$: 0.096 ipc: 2 - 12 - 25	Fixed scaling (symmetric) top-5: 0.519 $\cos_{\text{src}}$: 0.0644 $\cos_{\text{pc}}$: 0.187 ipc: 2 - 12 - 25	Fixed scaling (stretch) top-5: 0.440 $\cos_{\text{src}}$: 0.0861 $\cos_{\text{pc}}$: 0.216 ipc: 2 - 12 - 25	HSV top-5: 0.384 $\cos_{\text{src}}$: 0.0787 $\cos_{\text{pc}}$: 0.225 ipc: 2 - 12 - 25

Table 8: ILSVRC-based domain shifts we consider in our experimental setup.

Results. Quantitative results for all subsampling ratios and target domains are reported in Table 9, comparing flex-tuning and its variants to the fine-tuning baselines. In all settings, fine-tuning all layers performs badly as the network tends to overfit due to the rather small sample sizes. Furthermore, flex-tuning and its variants consistently outperform the other fine-tuning baseline, `ft-fc`. In the YUV and HSV settings, all three flex-tuning variants perform similarly as they indeed pick the same unit to fine-tune (second or third convolutional layer). The unit choice is not as consistent for the geometric transformations, but the actual classification accuracies are also similar for all flex-tuning variants.

Moreover, we observe that the best performing method is prep-tuning (*i.e.*, flex-tuning augmented with a pre-processng module). In fact, as was the case in Section 2, the specialized pre-processing modules performs very well to recover the original source domain on these transformations, even in the small sample size setting. This can also be observed in Table 10 and Figure 8, in whic we report quantitative and qualitative results for the pre-processing module experiments. For geometric transforms, we use a Spatial Transformer Network as described in Section 2, this time with a 4-layers encoder that takes as input 224x224 images. For color channel conversions, we use a simple network composed only of 1x1 convolutions, either with only one layer, which is enough to model linear relations such as RGB $\rightarrow$ YUV, or with 2 layers and tanh activation in-between for the more complex RGB $\rightarrow$ HSV transformation.

ILSVRC	flex				ft-		
	flex	fast	faster	prep	fc	ss	all
YUV (0.84)	0.856 ± 0.00	0.857 ± 0.00	0.830 ± 0.00	**0.902 ± 0.00**	0.775 ± 0.00	0.574 ± 0.00	0.817 ± 0.00
HSV (0.38)	0.750 ± 0.01	0.750 ± 0.01	0.750 ± 0.01	**0.895 ± 0.00**	0.422 ± 0.01	0.374 ± 0.01	0.582 ± 0.00
Scaling (stretch) (0.44)	0.626 ± 0.01	0.612 ± 0.02	0.596 ± 0.00	**0.780 ± 0.00**	0.361 ± 0.01	0.425 ± 0.00	0.595 ± 0.01
Scaling (sym.) (0.52)	0.703 ± 0.01	0.700 ± 0.00	0.700 ± 0.00	**0.837 ± 0.00**	0.531 ± 0.00	0.525 ± 0.00	0.659 ± 0.00
Rotation (0.74)	0.771 ± 0.00	0.769 ± 0.01	0.750 ± 0.01	**0.837 ± 0.00**	0.621 ± 0.00	0.499 ± 0.01	0.718 ± 0.00

(a) Data subsampling ratio 0.04 (∼ 1 image per class). Metrics are reported for 20 repetitions of the experiment with different sampled training images.

ILSVRC	flex				ft-		
	flex	fast	faster	prep	fc	ss	all
YUV (0.84)	0.893	0.893	0.893	**0.903**	0.835	0.699	0.808
HSV (0.38)	0.856	0.856	0.856	**0.905**	0.533	0.646	0.687
Scaling (stretch) (0.44)	0.724	0.696	0.696	**0.803**	0.502	0.584	0.653
Scaling (sym.) (0.52)	0.770	0.757	0.757	**0.856**	0.663	0.650	0.716
Rotation (0.74)	0.826	**0.832**	0.812	0.826	0.667	0.652	0.771

(b) Data subsampling ratio 0.5 (∼ 12 images per class).

ILSVRC	flex				ft-		
	flex	fast	faster	prep	fc	ss	all
YUV (0.84)	0.897	0.897	0.897	**0.903**	0.839	0.716	0.818
HSV (0.38)	0.863	0.863	0.863	**0.904**	0.545	0.676	0.670
Scaling (stretch) (0.44)	0.750	0.706	0.706	**0.778**	0.515	0.597	0.675
Scaling (sym.) (0.52)	0.787	0.770	0.770	**0.880**	0.668	0.655	0.728
Rotation (0.74)	**0.837**	0.829	0.812	**0.837**	0.674	0.663	0.764

(c) Experiments using the full target domain training dataset (∼ 25 images per class).

Table 9: Quantitative results for the ILSVRC12 source domain. We report on experiments with the proposed *flex-tuning* (`flex`), its two faster variants *fast flex-tuning* (`fast-flex`) and *even faster flex-tuning* (`faster-flex`), as well as flex-tuning augmented with a *preprocessing module* (`flex-prep`). We compare against transfer learning baselines: fine-tuning *all layers* in the architecture (`ft-all`), versus fine-tuning only the single last *fully-connected* layer (`ft-fc`).

ILSVRC	Ratio 0.010		Ratio 0.500		Ratio 1.000	
	prep (1)	prep (2)	prep (1)	prep (2)	prep (1)	prep (2)
YUV	**0.902**	0.901	**0.903**	0.711	**0.903**	**0.903**
HSV	**0.895**	0.686	0.896	**0.905**	0.895	**0.904**
Scaling (stretch)	**0.780**	-	**0.803**	-	**0.778**	-
Scaling (sym.)	**0.837**	-	**0.856**	-	**0.880**	-
Rotation	**0.837**	-	**0.826**	-	**0.831**	-

Table 10: Quantitative results from the pre-processing module experiments for different architectures. For the color channel experiments, `prep` (n) designates a pre-processing architecture with n 1x1 convolutional layers.

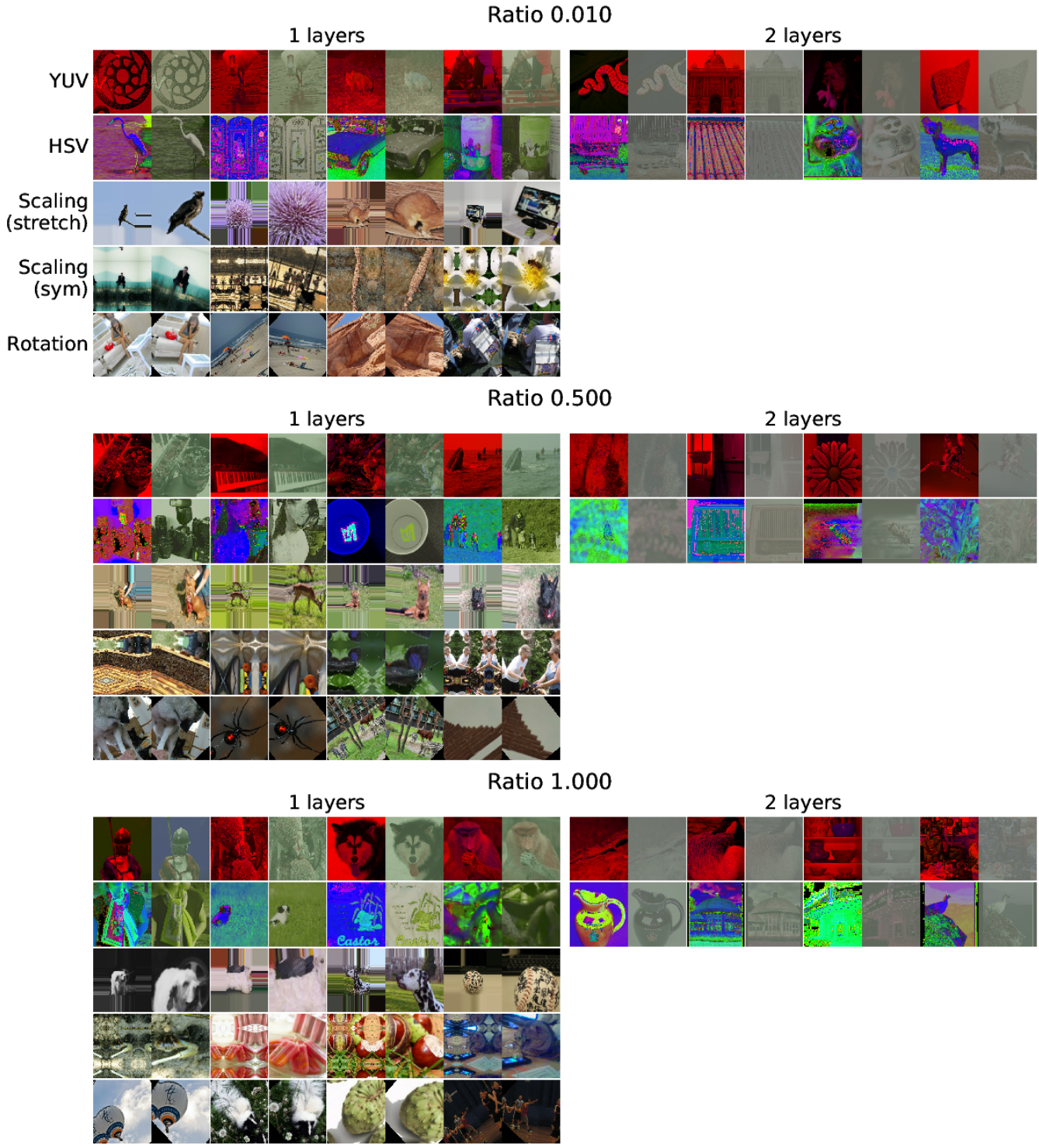


Figure 8: Qualitative results from the pre-processing module for each subsampling ratio, target domain and pre-processing module architecture. Figures read row-wise, two-by-two, where each pair contains the input image on the left (target domain) and the generated pre-processed image on the right (source domain). Best seen in PDF with zoom.

5. PACS Experiments

Datasets and base architecture. Our most challenging setting is based of the PACS dataset [10], initially introduced for domain generalization. The dataset contains 7 object categories, with occurences in 4 different artistic styles. We first randomly split each artistic style in a training, validation and test set. We then use a pre-trained ILSVRC12-pretrained Inception2 network as our base source model. We map PACS classes to the closest category in the ILSVRC classification task, and ignore the class "person" as it does not have any satisfying equivalent. For reference, even though it was not trained on it, it reaches top-1 accuracy of **0.885** on the test set for the **photo** split of PACS. We use the remaining artistic styles, **artistic paintings**, **cartoon** and **sketch** as our target domains.

			
Source domain	PACS (art)	PACS (cartoon)	PACS (sketch)
ILSVRC [13]	`top-1`: 0.532	`top-1`: 0.346	`top-1`: 0.142
1M images, 1000 classes	$\texttt{cos}_{\text{src}}$: 0.061	$\texttt{cos}_{\text{src}}$: 0.1361	$\texttt{cos}_{\text{src}}$: 0.211
Inception2	$\texttt{cos}_{\text{pc}}$: 0.090	$\texttt{cos}_{\text{pc}}$: 0.189	$\texttt{cos}_{\text{pc}}$: 0.257
`top-5`: 0.918	`ipc`: 2-20-200	`ipc`: 2-20-200	`ipc`: 4-40-400

Table 11: PACS-based domain shifts we consider in our experimental setup.

As for other datasets, we also create subsampled versions of each target training dataset by randomly sampling images with ratios 0.01, 0.1 and 1.0. A summary of our experimental setup is given in Table 11.

Results. Quantitative results comparing `flex` and variants to the fine-tuning baselines for all subsampling ratios and target domains are reported in the main paper. As for the ILSVRC setting, `ft-all` is very prone to overfitting and usually performs badly. Out of the other methods, `flex` and variants yields better performance than the other fine-tuning baseline, `ft-fc`, showing that there is benefit to selecting the best unit to tune.

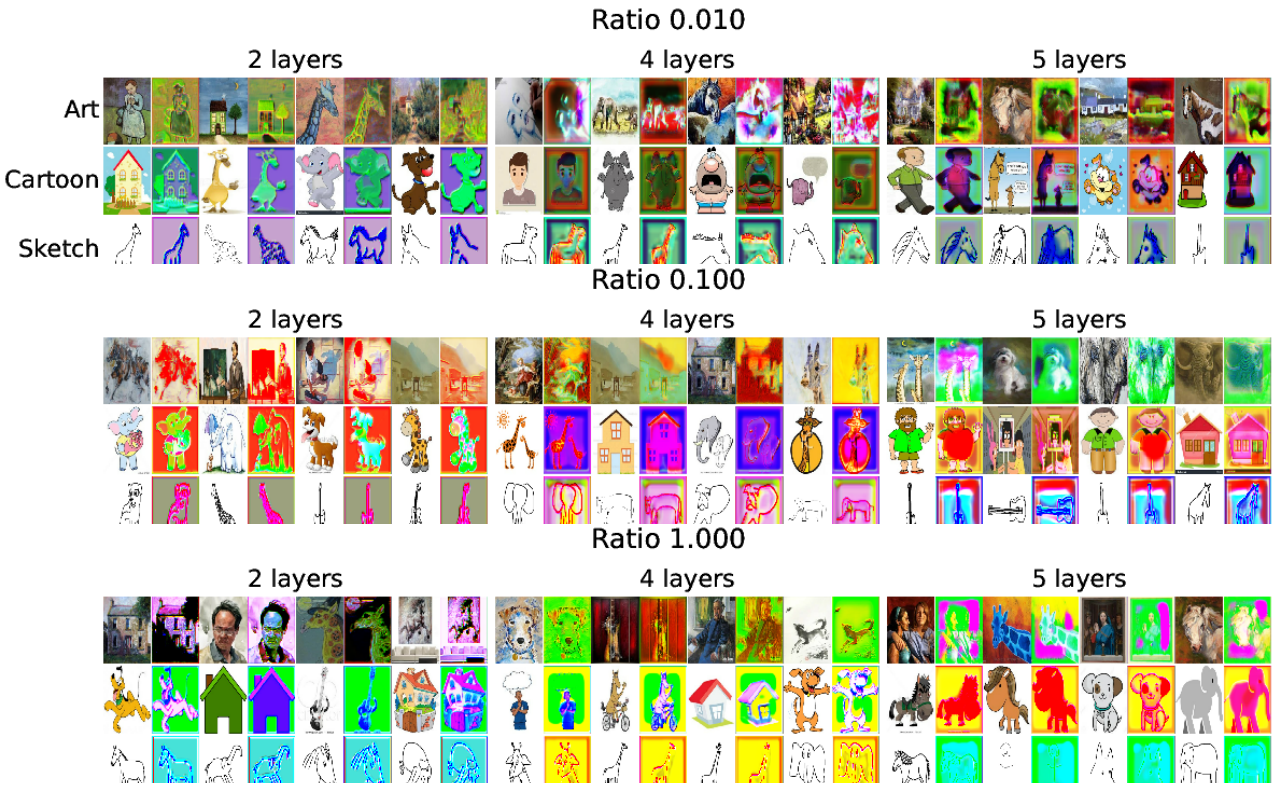


Figure 9: Qualitative results from the pre-processing module for PACS experiments.

Secondly, we note that in this scenario, the pre-processing module performs very badly and does not help flex-tuning. In fact, with these widely different artistic styles, the image-to-image translation problem becomes significantly harder than the classification task we actually want to solve. In consequence, prep-tuning always reduces to flex-tuning in that scenario.

This can also be seen in Table 12 and Figure 9, in which we report quantitative and qualitative results for the pre-processing module experiments. We used pix2pix as described in Section 2 with increased depth to match the larger input size.

ILSVRC	Ratio 0.010			Ratio 0.100			Ratio 1.000		
	prep (5)	prep (4)	prep (2)	prep (5)	prep (4)	prep (2)	prep (5)	prep (4)	prep (2)
Art	**0.365**	0.331	0.343	**0.504**	0.451	0.472	0.482	0.513	**0.568**
Cartoon	0.333	**0.369**	0.358	**0.493**	0.335	0.447	0.470	0.606	**0.633**
Sketch	0.361	0.354	**0.390**	**0.636**	0.422	0.286	0.551	0.531	**0.594**

Table 12: Quantitative results from the pre-processing module for PACS experiments. For the pix2pix network, `prep` (n) designates an architecture with n layers in the encoder and n layers in the decoder.

6. Retrieval Experiments

In this set of experiments, we assess how much the representations learned by the tuned network differ from the initial representations from the source network on the source domain: Since the target domain visually differs from the source, extracting features from the pretrained source network directly, e.g. as is done with fine-tuning, leads to misaligned representations. While tuning intermediate units could help "mend" the representations and recovering an embedding space close to that of the original source network. To evaluate this, we use the following retrieval experiment: We extract features for the initial source validation domain through the source network, as well as for the target domain through the flextuned (best intermediate unit according to validation accuracy) or finetuned network. For each target sample, we then retrieve its top-k nearest neighbors in the source domain and consider them correctly retrieved if they share the same semantic class, and evaluate the average precision (AP@k). we report AP@1, AP@10, and AP@100 for our main experimental settings in Table 13. Qualitative results are also available in Figure 10 to Figure 13 .

The results follow the same global observations as other experiments: **(i)** Fine-tuning the last fully-connected layer is enough to learn a classifier but can not modify the feature extract part of the network, hence low retrieval results; **(ii)** allowing to tune intermediate units often recover representations close to the one of the source embedding. In particular, when the target domain differ from the source domain at a very low-level (e.g., color channels or noise), then targetting a single unit such as in flex-tuning usually yields better results. For the other domain shifts, it seems that flex-tuning is usually a better alternative on larger architectures, while for smaller architectures, fine-tuning all layers yield the best results.

Source	MNIST				CIFAR			ILSVRC		PACS		
Target	Blurry	Occl.	-M	SVHN	Blurry	Noisy	QDraw	YUV	HSV	Art	Cartoon	Sketch
`ft-fc`	0.32	0.67	0.44	0.15	0.27	0.55	0.23	0.86	0.45	0.62	0.45	0.35
`ft-all`	**0.91**	**0.84**	**0.74**	**0.58**	**0.58**	0.58	**0.65**	0.79	0.48	0.82	0.81	0.87
`flex`	0.73	0.80	0.57	0.34	0.44	**0.63**	0.45	**0.94**	**0.95**	**0.90**	**0.84**	**0.90**

a) AP@1 results

Source	MNIST				CIFAR			ILSVRC		PACS		
Target	Blurry	Occl.	-M	SVHN	Blurry	Noisy	QDraw	YUV	HSV	Art	Cartoon	Sketch
`ft-fc`	0.38	0.70	0.47	0.19	0.36	0.58	0.27	0.80	0.47	0.66	0.53	0.39
`ft-all`	**0.91**	**0.85**	**0.76**	**0.63**	**0.64**	0.64	**0.69**	0.75	0.52	0.80	0.83	0.85
`flex`	0.76	0.83	0.61	0.40	0.52	**0.67**	0.54	**0.85**	**0.85**	**0.88**	**0.86**	**0.86**

b) AP@10 results

Source	MNIST				CIFAR			ILSVRC		PACS		
Target	Blurry	Occl.	-M	SVHN	Blurry	Noisy	QDraw	YUV	HSV	Art	Cartoon	Sketch
`ft-fc`	0.28	0.61	0.41	0.16	0.28	0.47	0.24	0.66	0.39	0.50	0.42	0.27
`ft-all`	**0.82**	**0.80**	**0.70**	**0.54**	**0.55**	0.53	**0.63**	0.61	0.42	0.65	0.70	0.66
`flex`	0.61	0.76	0.53	0.32	0.37	**0.56**	0.43	**0.71**	**0.72**	**0.75**	**0.74**	**0.71**

c) AP@100 results

Table 13: mAP retrieval results for the fine– and flex-tuned network embeddings of the target domain (second row), queried against the source domain embeddings (first row)

Figure 10: Qualitative Results for retrieval experiments on MNIST. First column contains the query from the target domain. Next groups of columns contains the ten closest samples from the source domain, when using the target embeddings from `flex`, `ft-fc` and `ft-all` respectively

Figure 11: Qualitative Results for retrieval experiments on CIFAR.

Figure 12: Qualitative Results for retrieval experiments on PACS.

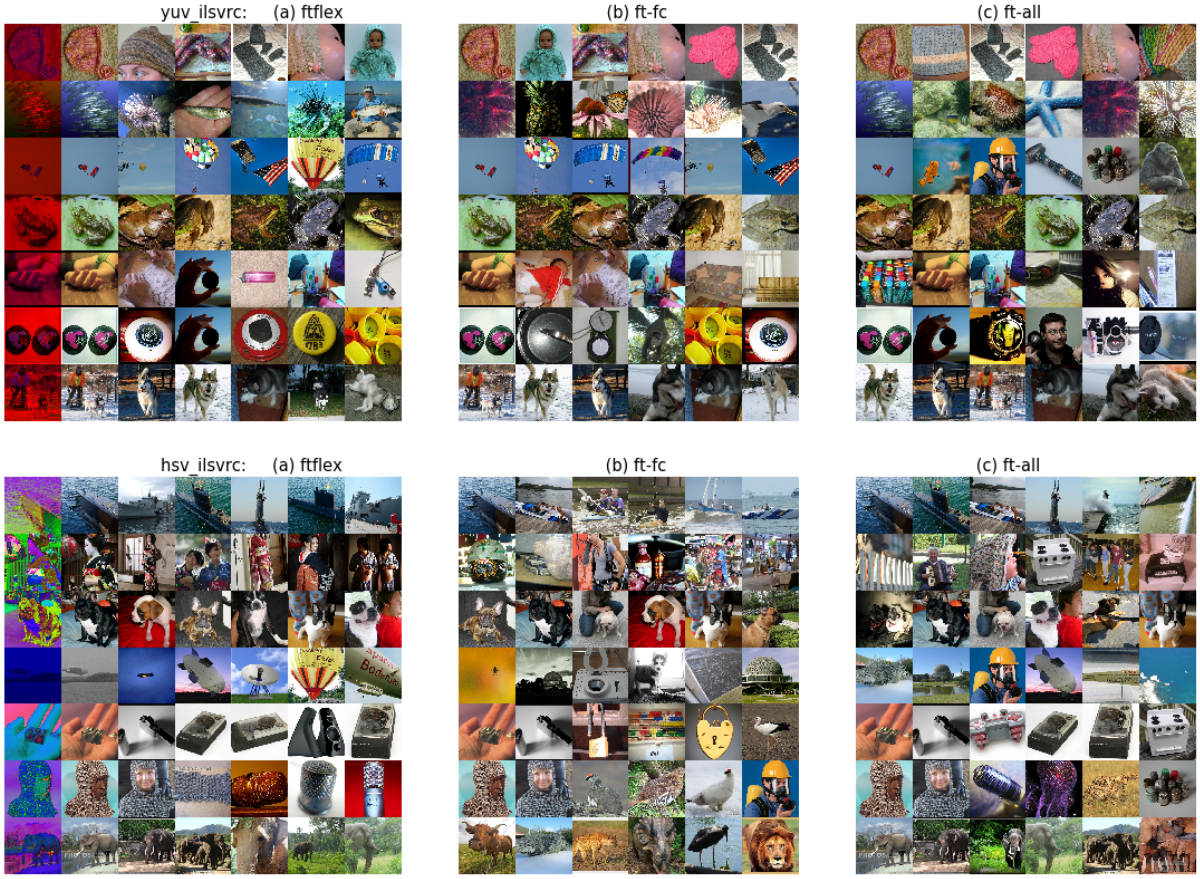


Figure 13: Qualitative Results for retrieval experiments on Colorized-ILSVRC.

7. Results for cosine classifiers

Finally in this section we report results for an additional transfer learning baseline inspired from [2]. It consists in applying fine-tuning on a neural network where the standard last linear classification layer is replaced by a cosine classifier, providing additional feature normalization. Note that we did not experiment on our ILSVRC-based settings as these would have required retraining the source network on ILSVRC's full training set. Nonetheless, in Table 14 we report results on the remaining setting and observe that (i) the base accuracy of the network trained with a cosine classifier is often slightly worse than the one trained without (see numbers in parenthesis in the first column) and (ii) this baseline performs much worse than simply finetuning a standard linear classification layer. A possible explanation is that since the method still only acts on the last layer, it is not enough to handle important visual dissimilarities at the input level. In particular, it aims to learn good "prototypes" in the feature space to describe each class, and in fact this method was rather intended to tackle different settings from ours: where the input domains are similar and semantic classes are different: Hence the feature representations of both domains lie in a similar subspace and only the prototypes have to be relearned for the new semantic classes.

	0.001 ratio		0.01 ratio		0.1 ratio		full	
	ft-fc	ft-cosine	ft-fc	ft-cosine	ft-fc	ft-cosine	ft-fc	ft-cosine
Blurry (0.63)	0.867	0.656	0.921	0.690	0.941	0.696	0.955	0.706
Occluded (0.58)	0.644	0.563	0.785	0.582	0.849	0.586	0.867	0.585
MNIST-M (0.35)	0.524	0.463	0.615	0.462	0.704	0.477	0.739	0.479
SVHN (0.20)	0.365	0.297	0.451	0.321	0.558	0.339	0.670	0.342

	0.001 ratio		0.01 ratio		0.1 ratio		full	
	ft-fc	ft-cosine	ft-fc	ft-cosine	ft-fc	ft-cosine	ft-fc	ft-cosine
Blurry (0.32)	0.408	0.253	0.444	0.334	0.525	0.338	0.560	0.341
Noisy (0.57)	0.555	0.508	0.583	0.585	0.602	0.586	0.633	0.586
QuickDraw (0.31)	0.359	0.259	0.475	0.308	0.569	0.315	0.581	0.390

Table 14: Results of the baseline using cosine classifiers similar to [2] on the MNIST and CIFAR settings

References

[1] S. Cheema, S. Gulwani, and J. LaViola. QuickDraw: Improving drawing experience for geometric diagrams. In *Conference on Human Factors in Computing Systems (SIGCHI)*, 2012.

[2] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang. A closer look at few-shot classification. In *International Conference on Learning Representations (ICLR)*, 2019.

[3] B. Chu, V. Madhavan, O. Beijbom, J. Hoffman, and T. Darrell. Best practices for fine-tuning visual classifiers to new domains. In *ECCV Workshop TASK-CV: Transferring and Adapting Source Knowledge in Computer Vision*, 2016.

[4] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learing (ICML)*, 2015.

[5] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research (JMLR)*, 2016.

[6] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[7] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. In *Conference on Neural Information Processing Systems (NIPS)*, 2015.

[8] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[9] Y. LeCun and C. Cortes. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/.

[10] D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales. Deeper, broader and artier domain generalization. In *International Conference on Computer Vision (ICCV)*, 2017.

[11] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *International Conference on Computer Vision (ICCV)*, 2001.

[12] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[13] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 2015.

[14] TensorNets. https://github.com/taehoonlee/tensornets.