Supplementary Material

A. Implementation details

A.1. Architectures

In our main experiments we use supervised and self-supervised pre-trained ResNets from [52] and [7] respectively. We implement the *expert* retrieval systems using a ViT-B/16 CLIP model [48] and a ResNet50 DINO model [7] (see Tab. 5 for a zero-shot evaluation on both the downstream fine-grained datasets as well as on the auxiliary datasets used in A). Note that the cost to search in the auxiliary dataset A is linear with its size and efficient approximate search methods exist [23, 27], however, when the number of images to search is $< 10M$ a brute force solution is still very fast since image embeddings can be stored in GPU memory and the search simply involves a matrix multiplication.

Moreover, for a fair comparison with [8], only in the TTA experiments, we follow [8, 35] and add a 256-dimensional bottleneck consisting of a fully-connected layer followed by a BatchNorm layer after the pre-trained backbone, and apply WeightNorm on the classifier. We consider the lower dimensional bottleneck as a projection layer and therefore drop the original projection heads used in MoCo [24] without any performance drop.

A.2. Memory bank

In this section we describe the memory bank introduced in Sec. 3.1.2. We use a memory bank as in [8, 24, 34] to allow for a larger set of negative examples without requiring more forward passes (and more GPU memory) and to store previously computed logits that are subsequently used to get "filtered" pseudo-labels.

We maintain a memory queue M of size m_b throughout training. M contains both the last layer features g of samples from T and A and the logits of samples from T according to the current model $f_{A|T}$.

$$M = \Big\{ g(t'(x)), f_{A|T}(t'(x)) \Big| x \in T \Big\} \cup \Big\{ g(t'(x')) \Big| x' \in A \Big\} \tag{5}$$

where t' is a weak augmentation, T the target dataset and A the auxiliary external dataset. Note that we do not keep track of pseudo-labels (or logits) in A since we do not assume that A contains the same label space of T. M is initialized with features and probabilities of m_b randomly selected target samples. And, at each mini-batch we update M by enqueue and dequeue similar to [8, 24, 34], where the momentum encoder is used with $m = 0$. The memory bank is updated on-the-fly with the current mini-batch and, together with features and logits. We also keep track of the unique image IDs that are used to aggregate logits and avoid using same image as negative samples. Furthermore, since our retrieval system R is fixed throughout training, for each sample in T (indexed by its unique ID) we associate a list of nearest neighbors in A that does not change and we use it to efficiently find the nearest neighbours present in M at each iteration.

B. Expert retrieval systems zero-shot evaluation

In this section we compare the zero-shot performance of our retrievers both on the fine-grained and the auxiliary datasets used to build A. First, for each dataset in the list of fine-grained/auxiliary datasets we compute image embeddings using an embedding model (ViT-B/16 or ResNet50 DINO) without using any data augmentation. This process is very fast, though it scales linearly with the dataset size, and its cost is essentially the cost of a single forward pass for each image to be indexed. Second, we store all the image embeddings in memory (this typically requires 100 times less memory than storing an image at 224×244 resolution) as well as their labels. Third, for each test image on a given dataset we compute its k-NNs from the list of embeddings and aggregate the corresponding labels (majority voting).

In Tab. 5, we compare the retrieval models with expert models that are trained on each dataset independently. Note that the CLIP model is strictly better than DINO in zero shot Top1 accuracy and it also competes with a fully trained model on multiple datasets. As we observed in Tab. 3 the stronger the retrieval the better the performance since more discriminative negative samples are used in the contrastive objective Equation (1).

Figure 6. **Visualization of samples retrieved from A.** *Top row:* First panel, distribution of CLIP similarity scores on A for the given query image. Third image, low CLIP score, forth image high similarity. *Bottom row:* 7-th closest image to the given query according to CLIP and DINO. Their top ranked images are often the same but DINO's ranking gets worse faster. Indeed, on the datasets composing A, DINO is a weaker zero-shot retrieval system than CLIP Tab. 5.

C. Tuning details

In this section we report the hyper-parameters that we use both for our training-time and test-time adaptation experiments. In particular, for train-time experiments we used

Table 5. **How expert are the retrieval systems?** Retrievals Top1 Accuracy (%) on the best number of k-NNs. We treat the number of NNs as a hyper-parameter and report the highest accuracy results where k is selected on a small validation dataset (10% of the training dataset). Furthermore, we report the accuracy of strong experts models ResNet50 (RN50) and ResNet101 (RN101) trained independently on each dataset of the external pool [18].

Dataset	CLIP	DINO	Expert [18]
External data pool			
ImageNet1k	72.5	73.1	**77.5** (RN101)
iNaturalist	41.3	38.1	**75.4** (RN101)
Food101	**89.1**	67.1	88.0 (RN101)
NWPU-RESISC 45	93.0	88.3	**96.5** (RN101)
Logo 2k	**83.8**	35.6	78.5 (RN101)
Fine-grained datasets			
Stanford Cars	72.3	21	**93.4** (RN50)
Stanford Dogs	70.9	68.4	**92.0** (RN50)
CUB200	68	67.8	**78.3** (RN50)
MIT67	**86**	71.6	78.9 (RN50)
FGVC-Aircrafts	45.5	36	**85.4** (RN50)

the standard 80/20 train/val splitting, and for test-time experiments we follow [8].

C.1. Train-time model adaptation with retrieval

In our train-time experiments we consider a labelled dataset S and a large auxiliary dataset of images A (see Sec. 4.1). The goal is to adapt a generic pre-trained model to the downstream labelled task S. The performance is evaluated on held out data T that is not used for further adaptation. This mimics the typical model customization scenario (transfer learning [1, 18]) solved with supervised fine-tuning. We pick S to be a labelled fine-grained classification dataset from the ones listed in Sec. 4.1.

In this scenario, we optimize our models[2] across different datasets[3] using SGD with momentum 0.9 and we use linear warm-up cosine annealing learning rate (we use 4 warm-up epochs, start learning rate 1e-5 and minimum learning rate 1e-6), other hyper-parameters are reported in Tab. 6. We fix m_b to $16k$ samples.

C.2. Test-time model adaptation with retrieval

In our test-time experiments we evaluate how well a given pre-trained model can adapt using an unlabelled dataset T. In particular, we are given a labelled dataset S that represents the downstream task and has the same label space as T but its covariates are shifted. As in previous experiments the auxiliary data pool A is taken as the concatenation of the datasets listed in Sec. 4.1. We evaluate downstream performance with Top1 accuracy on different domains for DomainNet-126 and across different classes for VisDA-C.

Also, in this case we train our models[4] on differently sized target datasets[5] use SGD with momentum 0.9 and we use linear warm-up cosine annealing learning rate (we use 4 warm-up epochs, start learning rate 1e-5 and minimum learning rate 1e-6), other hyper-parameters are reported in Tab. 6. Since both target datasets for TTA are larger than the fine-grained used in our train-time experiments, we increase the size of the memory bank to $64k$ samples. While, when working with 1% and 10% of T we use $m_b = 16k$.

D. Datasets details

Train-time adaptation and auxiliary datasets We choose both our fine-grained and the auxiliary datasets such that they cover different domains and are publicly available for download. Detailed data statistics are reported in Tab. 7.

Test-time adaptation datasets Following previous literature on test-time domain adaptation [8, 57, 65] we use VisDA-C [46] and DomainNet-126 [45] for evaluating our method on TTA and for comparing against baselines. Since DomainNet has noisy labels, we follow [8] and use a subset of it that only contains 126 classes from 4 domains (Real, Sketch, Clipart and Painting). We therefore evaluate our method on 7 domain shifts constructed from these 4 domains. Only for VisDA-C we compare the per-class top-1 accuracy, and then aggregate them by averaging.

E. Experiments design

In this section we further discuss the main motivations of our experimental study and the main baseline methods we used to evaluate $\mathrm{T}^3\mathrm{AR}$.

Fairness of comparison with existing adaptation methods. Our experiments are aimed at showing the value of using a new problem formulation for model adaptation that allows retrieval of external information. Hence, rather than aiming at comparing against other algorithms in similar settings, we use existing Train- or Test-Time algorithms to provide strong baselines to quantify what is the value of additional data for downstream adaptation. Our main contribution is to show that this setting can significantly improve accuracy, while still being widely applicable (*e.g.*, unlabelled

[2] For consistency, we keep the same hyper-parameters even when using the self-supervised pre-trained ResNet50 from [7].

[3] In case of Sup. FT 20%, to reduce the risk of over-fitting, we decrease the number of epochs to 30 and consider an halved batch size.

[4] In the TTA experiment, for consistency with the literature, we do not use backbones pre-trained with self-supervised objectives [7].

[5] In case of TTA on 1% and 10%, to reduce the risk of over-fitting, we decrease the number of epochs to 30 but do not reduce the batch size.

Table 6. **Detailed hyper-parameters configurations.** *Ref.* refers to the experiment (Table and adaptation method) where the model is mentioned. *Pre-tr. Arch.* describes the architecture and the pre-training objective used. *Pre-tr. data* refers to the pre-training data used to build the pre-trained architecture. *Target data* refers to the downstream classification dataset used for evaluation. For TTA it is the same as the pre-training dataset. *LR* is the base learning rate (with linear ramp-up and cosine decay and SGD with momentum 0.9). *WD* is weight decay. *MixUp* refers to the amount of data augmentation used, where 0. corresponds to no Mixup while 1. is the maximum amount of data augmentation allowed. *Batch Size* is the batch size considered (splitted across multiple GPUs, 8 Tesla V100).

Train-Time Adaptation								
Ref	Pre-tr. Arch.	Pre-tr. data	Target data	LR	WD	Mixup	Batch Size	Epochs
Tab. 1, Sup. FT [3]	Sup. RN50 [2]	IN1k	Stanf. Cars	0.1	0.01	0.1	1024	100
Tab. 1, Sup. FT [3]	Sup. RN50 [2]	IN1k	Aircrafts	0.1	0.01	0.1	1024	100
Tab. 1, Sup. FT [3]	Sup. RN50 [2]	IN1k	CUB200	0.1	0.01	0.	1024	100
Tab. 1, Sup. FT [3]	Sup. RN50 [2]	IN1k	MIT-67	0.1	0.01	0.1	1024	100
Tab. 1, Sup. FT [3]	Sup. RN50 [2]	IN1k	Stanf. Dogs	0.01	0.01	0.	512	100
Tab. 1, T^3AR [3]	Sup. RN50 [2]	IN1k	Stanf. Cars	0.1	1e-4	0.	1024	100
Tab. 1, T^3AR [3]	Sup. RN50 [2]	IN1k	Aircrafts	0.1	1e-4	0.	1024	100
Tab. 1, T^3AR [3]	Sup. RN50 [2]	IN1k	CUB200	0.1	1e-4	0.	1024	100
Tab. 1, T^3AR [3]	Sup. RN50 [2]	IN1k	MIT-67	0.1	1e-4	0.	1024	100
Tab. 1, T^3AR [3]	Sup. RN50 [2]	IN1k	Stanf. Dogs	0.01	1e-4	0.	512	100
Test-Time Adaptation								
Ref	Arch.	Pre-tr. data	Target data	LR	WD	MixUp	Batch Size	Epochs
Tab. 2, T^3AR [3]	Sup. RN50 [4]	DomainNet-126	DomainNet-126	0.1	1e-4	0.	1024	30
Tab. 2, T^3AR [3]	Sup. RN50 [4]	VisDA-C	VisDA-C	0.1	1e-4	0.	1024	30

Table 7. The number of training images, testing images and classes as well as the URL to download the dataset are listed below. The top part contains the auxiliary datasets in A, the middle part lists our fine-grained datasets and the bottom part contains test-time adaptation datasets.

Dataset	**Training Images**	**Testing Images**	**# Classes**	**URL**
NWPU-RESISC45 [14]	25,200	6300	45	https://www.tensorflow.org/datasets/catalog/resisc45
Food-101 [5]	75,750	25,250	101	https://www.tensorflow.org/datasets/catalog/food101
Logo 2k [58]	134,907	32,233	2341	https://github.com/msn199959/Logo-2k-plus-Dataset
iNaturalist [25]	265,213	3030	1010	https://github.com/visipedia/inat_comp
iMaterialist [41]	965,782	9639	2019	https://github.com/malongtech/imaterialist-product-2019
Imagenet [17]	1,281,167	50,000	1000	http://image-net.org/download
CUB-200 [56]	5994	5794	200	http://www.vision.caltech.edu/visipedia/CUB-200-2011.html
Stanford Cars [31]	8144	8041	196	https://ai.stanford.edu/~jkrause/cars/car_dataset.html
FGVC-Aircrafts [29]	6667	3333	100	https://www.robots.ox.ac.uk/~vgg/data/fgvc-aircraft/
CUB200 [29]	5994	5794	200	https://www.vision.caltech.edu/datasets/cub_200_2011/
MIT-67 [29]	5360	1340	67	https://web.mit.edu/torralba/www/indoor.html
Stanford Dogs [29]	12000	8580	120	http://vision.stanford.edu/aditya86/ImageNetDogs
DomainNet-126 [8, 45]	142334	-	126	http://ai.bu.edu/M3SDA/
VisDA-C [46]	152397	55388	12	https://github.com/VisionLearningGroup/taskcv-2017-public

images with a wide domain coverage are readily available from web-scale datasets).

Train time experiments In the training time experiments we are given a pre-trained model on some pre-training data (pre-trained either with supervision or self-supervision), a labelled dataset S and an unlabelled target dataset T. The goal is to adapt the model so that its performance on T is high. This is the standard transfer learning [1, 18] setting. We therefore use supervised fine-tuning as strong baselines (with hyper-parameter search Tab. 6). However, note that T^3AR is allowed to leverage an auxiliary unlabelled dataset A to further improve adaptation. In general, supervised fine-tuning methods are not designed to exploit side information in A. In this case one can resort to semi-supervised techniques to leverage a large set of unlabelled data (*e.g.* FixMatch [51] or CoMatch [34]). However, these methods cannot be applied in this scenario, since they assume that the labelled dataset S and the unlabelled dataset A are sampled from the same distribution. In our setting, A is more general

Table 8. Classification accuracy (%) on 7 domain shifts of DomainNet-126. All methods use ResNet-50 backbone. Bold is the highest. Performance of methods from the literature are taken from the cited papers [8, 26, 28, 35]. In Tab. 2 we report the accuracy as the number of samples available in T decreases.

Method	Source-free	R→C	R→P	P→C	C→S	S→P	R→S	P→R	Avg.
MCC [26]	no	44.8	65.7	41.9	34.9	47.3	35.3	72.4	48.9
Source only	-	55.5	62.7	53.0	46.9	50.1	46.3	75.0	55.6
TENT [57]	yes	58.5	65.7	57.9	48.5	52.4	54.0	67.0	57.7
SHOT [35]	yes	67.7	68.4	66.9	**60.1**	**66.1**	59.9	**80.8**	67.1
AdaContrast [8]	yes	**70.2**	69.8	**68.6**	58.0	65.9	**61.5**	80.5	**67.8**
T^3AR (w/o retrievals)	yes	68.5	67.9	63.4	53.1	63.9	52.7	80.4	64.3
T^3AR (Ours)	yes	**70.2**	**70.0**	66.8	**60.9**	64.1	59.8	**81.0**	67.5

Table 9. Classification accuracy (%) on VisDA-C train → val. All methods use ResNet-101 backbone except the on-target rows, which use ResNet-18 as student network. Bold is the highest; underline is the second highest. Performance of methods from the literature are taken from the cited papers [8, 26, 28, 35]. In Tab. 2 we report the accuracy as the number of samples available in T decreases.

Method	source-free	plane	bcycl	bus	car	horse	knife	mcycl	person	plant	sktbrd	train	truck	Avg.
CAN [28]	no	97.0	87.2	82.5	74.3	**97.8**	**96.2**	90.8	80.7	**96.6**	**96.3**	87.5	**59.9**	**87.2**
MCC [26]	no	88.7	80.3	80.5	71.5	90.1	93.2	85.0	71.6	89.4	73.8	85.0	36.9	78.8
Source only	-	57.2	11.1	42.4	66.9	55.0	4.4	81.1	27.3	57.9	29.4	86.7	5.8	43.8
SHOT [35]	yes	95.3	87.5	78.7	55.6	94.1	94.2	81.4	80.0	91.8	90.7	86.5	59.8	83.0
+ On-target	yes	96.0	**89.5**	84.3	67.2	95.9	94.2	91.0	81.5	93.8	89.9	89.1	58.2	85.9
AdaContrast [8]	yes	97.0	84.7	84.0	77.3	96.7	93.8	91.9	84.8	94.3	93.1	**94.1**	49.7	86.8
+ On-target	yes	**97.2**	87.0	**86.7**	**81.7**	95.5	91.6	93.5	**86.6**	95.3	90.9	92.8	47.9	**87.2**
T^3AR (w/o retrievals)	yes	90.3	83.9	72.4	73.0	93.1	88.9	82.6	82.4	90.1	87.8	90.3	40.5	81.3
T^3AR (Ours)	yes	96.8	87.5	86.2	74.8	96.7	90.5	**93.8**	82.4	91.7	91.3	91.1	45.9	85.7

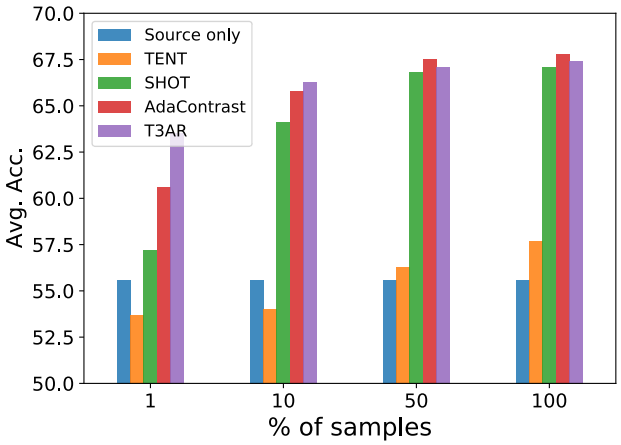

Figure 7. Classification Top1 Accuracy (%) of test-time adaptation methods on DomainNet-126 as the number of available target data T increases. We show that exploiting retrieved samples helps T^3AR especially in the low data regime.

and does not need to be sampled from the same distribution as S (A can contain many more concepts than the ones present in S). Therefore, a reasonable baseline is to allow a fine-tuning method to leverage A somehow. The easiest way to do this is by using the whole dataset A to pre-train the model. Since A is, in general, unlabelled, we use a self-supervised objective function (DINO [7]). Then, this pre-trained model is used as starting checkpoint for fine-tuning on S. In this way, the final adapted model contains in its weights both information on the samples from A and samples from S exactly as in the case of T^3AR . We call this adaptation strategy *Self-Sup. pre-training* on A and compare it with T^3AR in Tab. 10. Note that T^3AR outperforms this paragon. This suggests that, while pre-training a model on as many data as possible is a strong baseline, it is possible to further improve downstream performance by looking back at pre-training data after the downstream ones are available (this observation is aligned with empirical results in [54]).

Table 10. Comparison of T^3AR with a self-supervised model pre-trained on A and then fine-tuned on S. We observe that the average accuracy of T^3AR outperform the retraining paragon.

Dataset	Stanf. Cars	CUB200	MIT67
Self-Sup. pre-tr. on A	91.4	79.2	74.6
T^3AR	**93.0**	**80.3**	**75.9**

Test time experiments In the test time experiments we are given a pre-trained model on some labelled dataset S consisting of source data and an unlabelled target dataset T. The goal is to adapt the model to T (without using S). This is the standard test-time adaptation setting [8, 35, 57]. We therefore compare T^3AR with strong TTA baselines.

Furthermore, as typically done in the literature, we add a direct comparison with UDA methods [26, 28], these allow the method to look back at S once T is obtained. The results are reported in Tab. 2, Tab. 8 and Tab. 9.

Datasets subsampling To test the efficacy of external data both in train and test time experiments we tested our method and baselines using both full sized and subsampled downstream datasets. In particular, we subsampled each dataset using stratified sampling.

F. Detailed results on TTA experiments

In Tab. 8 and Tab. 9 we report Top1 accuracy on all the domains in DomainNet-126 and on all the classes in VisDA-C. We compare our method with state-of-the-art UDA and TTA methods.

In Tab. 8 our method outperforms MCC even though it has not access to the source datasets. Our method also compares favourably with TTA baselines, being behind only to AdaContrast on the entire dataset size but being the best when fewer samples are available during adaptation Tab. 2. Our method performs better than others when only 1% and 10% of the datasets are allowed for adaptation since it leverages external information from the retrieved samples. In fact, all other methods only rely on synthetic data augmentations to drive the learning process, and therefore, are not fully able to describe the complex target data manifold when data are limited. Interestingly, as more samples are allowed to be used, synthetic data augmentations seems to suffice and the performance of other methods gets increasingly better. We note that our method achieves the best performance on 3 out of 7 domain shifts and it is on par with AdaContrast on one (R→C).

In Tab. 9 we compare our method on the VisDA-C adaptation dataset. It gets the best accuracies on the *bcycl* class and outperforms AdaContrast (a strong TTA adaptation baseline [8]) on 4 our of 12 classes. Furthermore, Tab. 2 we show, once again, that our method compares favourably w.r.t. the baselines when few samples are allowed for adaptation.

Sensitivity to the number of retrievals In Fig. 8 we study the sensitivity of $\mathrm{T}^3\mathrm{AR}$ as the number of retrieved nearest neighbors increases. The x-axis represents the number of number of retrievals allowed per sample, with $NNs = 1$ we can retrieve as many samples as there are in the target dataset, with $NNs = 2$ twice its size, etc. We also report the performance of randomly retrieving as many samples as there are in the target dataset (diamond markers at $NNs = 0$). Our results show a diminishing return in performance as the number of NNs increases. Since retrieving more samples increases (linearly) adaptation time, our experiments suggest that a good trade-off, that holds across different datasets and allows to discount compute over marginal accuracy improvements, is to retrieve twice as many samples as the target datasets.

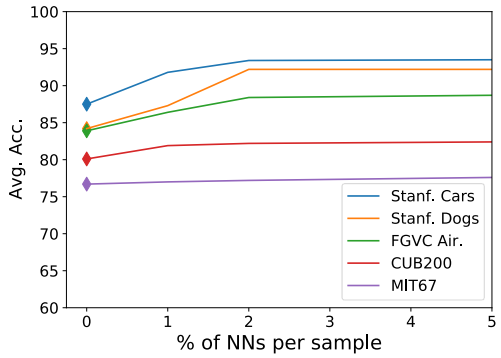

Figure 8. **Accuracy as a function of the number of the retrievals.** Classification Top1 Accuracy (%) of $\mathrm{T}^3\mathrm{AR}$ on train time adaptation on fine-grained classifications datasets as a function of the number of retrieved nearest neighbors. We denoted with diamonds the reference performance when random retrievals are used, in this case the number of retrievals is 1. Note that as the number of retrievals increases, as well as the adaptation time, $\mathrm{T}^3\mathrm{AR}$ saturates its performance around 2-5 retrievals across different datasets.

F.1. Main limitations

In our ablation studies we have showed that adding samples from A to adapt a downstream model leads to improved downstream performance on various adaptation benchmarks. Nonetheless, the user is responsible to bring in relevant data A (as relevant as possible to improve the contrastive loss on negative pairs) and to maintain A as it grows larger and larger. In practice, there is no bound on the size of A and even if similarity based retrievals are very fast, their throughput staturates as more samples are added. We leave to future work how to leverage fast approximate searches [23, 27] on large indexed databases and fast database re-indexing.