Train/Test-Time Adaptation with Retrieval

Luca Zancato Alessandro Achille Tian Yu Liu* Matthew Trager
Pramuditha Perera Stefano Soatto

AWS AI Labs

zancato@amazon.it
{aachille,tytianyu,mttrager,pramudi,soattos}@amazon.com

Abstract

We introduce Train/Test-Time Adaptation with Retrieval ($\mathrm{T}^3\mathrm{AR}$*), a method to adapt models both at train and test time by means of a retrieval module and a searchable pool of external samples. Before inference,* $\mathrm{T}^3\mathrm{AR}$ *adapts a given model to the downstream task using refined pseudo-labels and a self-supervised contrastive objective function whose noise distribution leverages retrieved real samples to improve feature adaptation on the target data manifold. The retrieval of real images is key to* $\mathrm{T}^3\mathrm{AR}$ *since it does not rely solely on synthetic data augmentations to compensate for the lack of adaptation data, as typically done by other adaptation algorithms. Furthermore, thanks to the retrieval module, our method gives the user or service provider the possibility to improve model adaptation on the downstream task by incorporating further relevant data or to fully remove samples that may no longer be available due to changes in user preference after deployment. First, we show that* $\mathrm{T}^3\mathrm{AR}$ *can be used at training time to improve downstream fine-grained classification over standard fine-tuning baselines, and the fewer the adaptation data the higher the relative improvement (up to 13%). Second, we apply* $\mathrm{T}^3\mathrm{AR}$ *for test-time adaptation and show that exploiting a pool of external images at test-time leads to more robust representations over existing methods on DomainNet-126 and VISDA-C, especially when few adaptation data are available (up to 8%).*

1. Introduction

While Deep Learning models are evolving rapidly, machine learning systems used in production are updated rarely, as each deployment requires the provider to engage in a complex process of scaling, securitization, certification of new model and dataset cards, bias evaluation, and regression tests. It is now common for users to adapt trained models to their specific use cases, or to the changed context as time goes by [16, 39, 60]. Such adaptation can be performed by fine-tuning on a specific dataset S owned by the user [1, 18]. However, on an even finer time-scale, users may want to adapt their models based on data they observe at test time, bypassing the time-consuming annotation process [8, 35, 53, 57]. Test-Time Adaptation (TTA) refers to the problem of adapting a source model to a target task T represented by test data, for which no ground-truth labels are given.

This trend is exacerbated by the advent of Foundation Models [2, 10, 38, 59], at least in the visual domain where tasks can be antagonistic and models are sensitive to even subtle changes in the data distribution. At the same time, both users and providers typically have access to ever-growing pools of auxiliary data, albeit often heterogeneous (pertaining to concepts other than the one of interest at test-

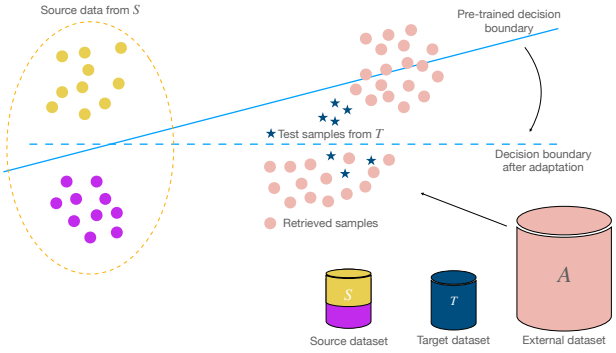

Figure 1. **Adaptation with retrieval from an external data pool.** Illustration of how $\mathrm{T}^3\mathrm{AR}$ exploits target data T and the external data pool A to adapt the decision boundary after pre-training on the source datasets S. For new test queries from the target dataset, $\mathrm{T}^3\mathrm{AR}$ approximates the local data manifold around T by retrieving similar unlabelled examples from A. Then, it updates the decision boundary with a contrastive self-supervised objective.

*Work done during an internship at AWS AI Labs.

time), and without annotations. Yet it seems plausible that, somewhere within these large pools of data, there may be information useful for the task at hand.

In this paper, we tackle the problem of performing test-time adaptation by retrieving information from a large, unlabeled, heterogeneous, and evolving dataset. The same procedure could also be followed by the provider, if they have access to auxiliary internal data and wish to adapt the production model based on trends observed in test data. We refer to our method as *Train/Test-Time Adaptation with Retrieval*, or $\mathrm{T}^3\mathrm{AR}$.

$\mathrm{T}^3\mathrm{AR}$, if solved, would enable a number of real-world tasks that have thus far frustrated practitioners. For instance, it would allow a user to select, among a vast data lake A, which samples to use for a training, based on labeled and unlabeled samples [61]. It would also enable nimble inference, by adapting a modest-size model to specific tasks, rather than relying on an unwieldy model to master all trades. Finally, it would enable *reversible adaptation:* While in the case of language models tasks are generally synergistic [44], in vision tasks can be antagonistic.[1] Therefore, a model adapted to one task may behave poorly on another, and a model that encompasses both would require significantly higher capacity [2, 10, 38, 59], to the detriment of inference efficiency. In $\mathrm{T}^3\mathrm{AR}$, changing the target data T changes the subset of the data pool A that is retrieved, with no impact on other models, instantiating smaller independent models for antagonistic tasks, rather than coercing them into a larger one, likely multiplying inference costs.

$\mathrm{T}^3\mathrm{AR}$ can be used in a continual setting, where at each time t one has a different target T_t, and the auxiliary task A is composed of the union of all prior targets $T_0, \ldots, T_t$. The retrieval system should automatically determine what information from whatever past targets is relevant to the present, and what information is redundant in A and can be eliminated. The important difference compared to ordinary continual learning is that each step starts with the base model, so there is no catastrophic forgetting, and what is updated is the auxiliary task. In other words, the integration of information occurs in A, not in the trained model f.

1.1. Related problems

$\mathrm{T}^3\mathrm{AR}$ relates to unsupervised domain adaptation (UDA) [26, 28, 45], since the target dataset is not annotated. However, in UDA one assumes that the source dataset S is available along with the target T, which is not necessarily the case in $\mathrm{T}^3\mathrm{AR}$ since users may want to bypass annotation altogether, and directly adapt the pre-trained model using the auxiliary dataset A, based on the target task T, without having direct access to S.

[1] *E.g.*, localization requires marginalizing identity, whereas recognition requires marginalizing location, making the features that are informative for one detrimental to the other [2, 38].

$\mathrm{T}^3\mathrm{AR}$ also relates to semi-supervised learning (SSL) [32, 34, 51], since the target dataset T and the auxiliary dataset A are not annotated. However, in SSL one assumes that labeled S and unlabeled data are drawn from the same joint distribution, which is not the case for T and A in $\mathrm{T}^3\mathrm{AR}$, and, in any case we do not aim to infer labels of A, and just use it to improve the model on the target task.

$\mathrm{T}^3\mathrm{AR}$ is also related to open-set domain adaptation [6, 49], since the auxiliary dataset A is heterogeneous and does not share the same label space as the source and target task. It is also related to out-of-distribution detection (OOD) [20, 62], since one needs to decide whether to add samples from the auxiliary dataset, and to active learning [50], since one needs to decide what samples to add.

Naturally, $\mathrm{T}^3\mathrm{AR}$ closely relates to test-time adaptation (TTA) [8, 35, 53, 57, 65], and to memory-augmented or retrieval-based architectures [3, 11, 36], widely developed in the language domain [4, 33, 63], where the hypotheses live in the same space of the data and nuisance variability is limited to paraphrasing.

In summary, $\mathrm{T}^3\mathrm{AR}$ lies at the intersection of UDA, SSL, OOD, TTA, Active Learning, and Retrieval, yet it does not fit neatly into any of them, making both the survey of related literature (Sect. 2) and experimental assessment (Sect. 4) non-straightforward.

1.2. Key ideas and contributions

We propose a method to solve $\mathrm{T}^3\mathrm{AR}$, based on a target unlabeled dataset T, that selects samples from an auxiliary dataset A, using a retrieval model R.

Starting from any model f_S pre-trained by the provider on a dataset D and later fine-tuned by the user on a labelled dataset S, our method finds subsets of an auxiliary dataset A that are relevant for the target dataset T, using nearest neighbors in A to samples in T, measured in a representation space computed by a retrieval model R (in our case, a CLIP embedding [48]).

The key technical contribution is a contrastive loss used for updating the model f_S to a new model $f_{A|T}$, whereby negative pairs are selected by retrieving samples from the external dataset A that are *informative* of T using the retriever R. Furthermore, to improve training stability, we exclude same-class negatives pairs from T by exploiting assigned pseudo-labels obtained by averaging predicted logits on different data augmentations. Our method can be thought of as a form of contrastive "dataset augmentation" by enlarging the user data with samples drawn from a different (unlabeled) dataset A, based on guidance provided by a retriever R. This procedure can be followed by both the user and the provider, thus empowering them to adapt the core model (train-time adaptation) or a sequence of disjoint custom models (test-time adaptation).

We show that applying $\mathrm{T}^3\mathrm{AR}$ improves downstream

classification accuracy over the paragon supervised fine-tuning [1, 18] for train-time and test-time adaptation methods [8, 35, 57] for test-time. In particular, as the number of data available during adaptation decreases, $\mathrm{T}^3\mathrm{AR}$ improves by up to 13% and 8% in relative Top1 accuracy at train and test time, respectively.

2. Related work

As we anticipated in the introduction, the problem we tackle has close connections with a number of areas of investigation in machine learning, including UDA, SSL, OOD, TTA, Active Learning, and Retrieval.

UDA and TTA Unsupervised Domain Adaptation (UDA) has a long history and it has been explored in a variety of different visual tasks, image classification [26, 28, 45], object detection [13] and semantic segmentation [55]. The main goal of UDA methods is to reduce the performance drop of pre-trained models when deployed on shifted target domains without using any target annotation. One of the most successful ideas in UDA literature is source and target feature space alignment. For example, [37] exploits Maximum Mean Discrepancy, [45] leverages a multi-source moment matching objective, [26] uses a non adversarial reduction of the class confusion and [28] employs a contrastive adaptation objective to model intra-class and inter-class domain discrepancy. However, all these methods require knowledge of the target distribution before model deployment, which highly limits their applicability in the wild. On the other hand, typical test-time adaptation (TTA) methods only use the target dataset during adaptation [8, 57] and usually no modification to the pre-training loss is allowed (a notable exception is [53]). Therefore, test-time adaptation is carried out exploiting regularities between source S and target data T. For example, it is often assumed that the target data shares the same class distribution with the source one, or that the un-adapted decision function is not far from the target [57]. Under these assumptions, [57] minimizes the entropy of the predictions to quickly adapt a given pre-trained model. [65] takes this approach one step further and exploits different synthetic data augmentations to further improve performance. Among other test-time adaptation methods, AdaContrast [8] is the closest to our solution since it leverages a contrastive loss for adaptation. However, as in previous methods, only synthetic data augmentations are used to construct the self-supervised contrastive loss. On the other hand, our method is not bounded to synthetic data augmentations and augments samples in T by leveraging other real data to better capture the variability in T.

While $\mathrm{T}^3\mathrm{AR}$ is close to TTA [8, 35, 53, 57], it differs in that we expect that the dataset used for adaptation is not just T, which is assumed to share the same label space of S, but also A, a typically very large dataset largely irrelevant to T. Hence, we leverage a retriever R to find the needles in the haystack, an element not present in the TTA literature.

Retrieval/memory augmented models Recently, retrieval based models have been used to solve symbolic manipulation [22], anomaly detection [21], image generation [11] and image classification [36]. In particular, [36] shows that augmenting a standard image classification model with an explicit image retrieval module highly improves accuracy on long tailed classification datasets. [11], instead, uses retrieved images as guidance for generating highly detailed uncommon concepts. Retrieval based models have also found applications on other domains other than Computer Vision. For example, in the NLP domain, several recent methods leverage large corpora to augment pre-trained large language models predictions with a non-parametric memory module [4, 12, 33, 63]. In particular, [4] shows that augmenting a large pre-trained language model with an external indexable database has mainly one advantage: higher performance w.r.t. the number of deployed parameters, which in turn unlocks the use of smaller/faster models that are less likely to memorize the training data. However, this result has yet to be reliably verified for large scale computer vision models. One of the main reasons for this discrepancy is that in the language domain the query and the data/representation live in the same space, so the answer to a query, expressed as a string of text, is a string of text which may potentially be in the knowledge base or easily interpolated from it. However, in the image domain it is usually not reasonable to assume that the answer to a given query already exists in some indexable database or knowledge base (*e.g.* downstream labels might differ from pre-specified labels in the knowledge base or database). Hence, in our case, we do not assume the auxiliary dataset A has ready answers to our queries.

3. Method

We assume that there is a *provider* who pre-trains a model g on a dataset D obtaining $g_D : X \to Z$ where X are RGB images and $Z = \mathbb{R}^d$ where d is the dimension of the feature space. Here, D is a large dataset which is typically not accessible after pre-training and may, with time, become obsolete.

A *user* has access to g_D, but wishes to improve it on a specific dataset S to build a custom classifier $f_S : X \to Y$, using g_D as a backbone, and fine-tuning it along with a linear layer.

Test data owned by the user and optionally made available to the provider, is drawn from an unlabeled dataset T, which may be different from both S and D, but shares the same hypothesis space Y of S [8,53,57]. In particular, there may be a domain shift from S to T, or the two may pertain to entities, such as products or fashion, that evolve over

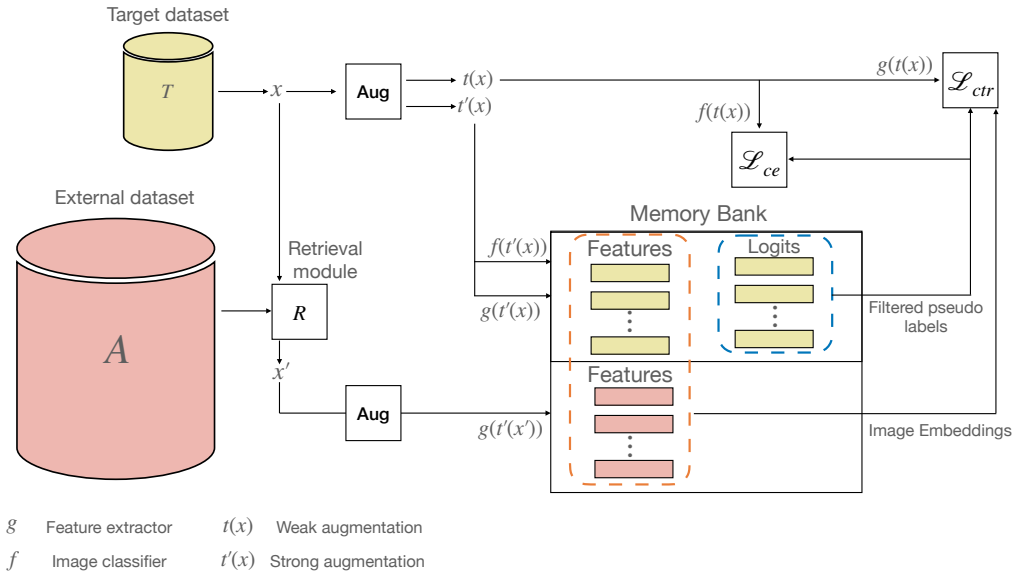

Figure 2. **Our proposed framework.** Given an image x from the unlabeled dataset T, an auxiliary external dataset A, and a retrieval method R, T^{3}AR strongly augments x with t' and stores the logits and features in a memory bank. Moreover, for each image x in T an independent retrieval system R (*e.g.* CLIP) retrieves from an external data pool A a given number of *related* images that are strongly augmented and saved as features into the same memory bank (logits of retrieved images are discarded). At each iteration, for each image x "filtered" pseudo-labels are generated leveraging logits in the memory bank. Pseudo-labels are then used both as targets to train the class predictor on weakly-augmented images with t' and as supervision for the contrastive loss. The contrastive loss is computed following instance discrimination on the augmented views of the same image against embeddings of images with different pseudo-labels and embeddings retrieved from the auxiliary pool of data A.

time. If such out-of-distribution phenomenon is manifest in the test samples, not only g_D, but even f_S will perform poorly.

The goal of T^{3}AR is to train an adapted model, leveraging an auxiliary dataset A, starting from f_S, but without directly accessing S, and leveraging instead unlabeled data from T available at inference time. We call the resulting adapted model $f_{A|T}$, where the dependency on S is implicit in its pre-training. We note that A may be a private dataset, accessible to the user but not the provider. Conversely, the provider may have an internal dataset that may be available to adapt the model to commonly observed tasks from opted-in users, using the same process followed by the user to perform test-time adaptation.

The goal of T^{3}AR is to train a model $f_{A|T}$ that improves the baseline f_S and gets as close as possible to the paragon which is to train with the entirety of the datasets D, S, A and T. To this end, we consider any generic pre-training g_D and fine-tuning f_S, and perform **retrieval** by finding subsets of A that are informative of T. We do so by finding nearest neighbors of samples of T in A using CLIP embedding space computed by a retrieval model R. We then perform **aggregation** to combine T with A into an "augmented training set". This is the key technical contribution of our work and is implemented as follows. Given each datum in $x \in T$, we create multiple augmentations x_i, and use f_S to compute the corresponding pseudo-labels. Then, we consider a contrastive loss whose objective is to pull closer features of different views (positive pairs) while pushing away features of different images (negative pairs). We consider as negatives the retrieved samples from A that are neighbors to T with R and samples in T with negative pseudo-labels.

T^{3}AR finds samples in A that are synergistic with T as contrastive neighbors and avoids pushing away same-class pairs to learn better semantically meaningful clusters. Redundant information is avoided simply by removing samples with near-duplicate embeddings according to R [3, 4].

Note that the contrastive loss does not update the linear classifier but only the features. So, T^{3}AR updates the classifier $f_{A|T}$ with supervision from pseudo-labels generated from T [8, 34, 51].

We now describe each component of method in detail.

Retrieval module Since the search cost scales with the data pool size any slow retrieval algorithm is not a feasible solution [42, 61]. We therefore use a fast retriever R whose main goal is to filter irrelevant data in A given target samples in T (*e.g.* out-of-distribution or near duplicate).

The retrieval module consists of a general image encoder $R : X \to \mathbb{R}^d$, that we use to index the auxiliary pool of

images A. We note that different retrieval systems lead to largely different retrieval distributions that mostly depend on the invariance classes imposed during the retrieval pre-training objective. For example, a CLIP model [48] is trained to match images with likely captions, while a self-supervised model (*e.g.* DINO [7]) is trained to be invariant to per-sample synthetic data augmentations. We evaluate the impact of the retrieval choice in Tab. 3. Differently from [4, 8], we do not use specialized fast approximate nearest-neighbor search (such as FAISS [27] or SCaNN [23]). Instead, we simply use a brute force search on the most similar keys (embedding indexes), since when the size of the external database does not exceed $10M$ the time reduction of approximate nearest neighbor search is minor.

Encoder initialization Since $\mathrm{T}^3\mathrm{AR}$ does not impose any restriction on the objective used to pre-train g_D, we shall consider models pre-trained both with a supervised [18] or self-supervised [7] objective (see Tab. 1).

3.1. Learning objective

Our learning objective consists of two parts. First, a self-supervised objective function that is used to incorporate retrieved information from A both at train and test time (see Sec. 3.1.1). Second, a cross-entropy objective that is driven by ground truth labels at training time and by pseudo-labels at test time (Sec. 3.1.2).

3.1.1 Retrieval-augmented objective function

In this section we describe the self-supervised objective function that we exploit to incorporate information from A. Inspired by recent advances in self-supervised objective designs [7, 9, 24] we exploit a contrastive objective driven by pairwise information. In particular, we follow the *instance-discrimination principle*: features of different views of the same image (positive pairs) are pulled closer, while features of different images (negative pairs) are pushed away. The key insight is that, even in presence of domain shift, the contrastive loss discriminative power increases with more negative samples [9, 30]. However, adding easily separable negatives does not provide much learning signal. We therefore use the retrieval module R to modify the noise distribution and gather images that serve as *harder negatives*.

Retrieval-augmented contrastive loss As in [8], given an image x, we create a weakly augmented view $t(x)$ and a strongly augmented view $t'(x)$. Then, we apply the InfoNCE loss on $q = g(t(x))$, $k = g(t'(x))$ and the set of strongly augmented negatives $\mathcal{N}_q$. Here, g denotes the last layer features (before the classifier head) extracted by the model $f_{A|T}$ being trained and the set $\mathcal{N}_q \subset A \cup T$ is composed of different-class samples from T and by retrieved samples from A (nearest neighbors of x according to R).

$$\mathcal{L}_{\text{ctr}}(x) = -\log \frac{\exp(q \cdot k/\tau)}{\sum_{j \in \mathcal{N}_q} \exp\left(q \cdot k_j/\tau\right)} \tag{1}$$

where τ is a temperature hyper-parameter and all k_j are feature embeddings stored in a memory bank of length n_p that is updated by appending the new embedding k at each step [8, 24].

As observed in [8, 9, 24, 30], the InfoNCE loss in Eq. (1) might strive to minimize the cosine distance between q and k while maximizing the cosine distance of q and all the negatives in the denominator. In particular, not pushing away same-class pairs helps in building a feature space that is more aligned with the semantic of the downstream task. Therefore, when the label information (or pseudo-labels) is available, we modify $\mathcal{N}_q$ not to include samples with the same label y (or pseudo-label $\bar{y}$) of x:

$$\mathcal{N}_q^{\text{lab}} := \{j \mid y \neq y^j\} \cup \emptyset \tag{2}$$

In Sec. 3.1.2 we describe how to compute pseudo-labels $\bar{y}$ on the target set T.

Furthermore, we leverage the auxiliary data available in A to increase the number of negatives. However, as observed in [54] naively leveraging a large pool of uncurated data in a self-supervised contrastive loss might not lead to performance improvements since negatives can be less informative (easy negatives). To overcome this limitation we leverage the retrieval system R whose task is to gather more relevant negatives (*hard negatives*). More specifically, we build $\mathcal{N}_q^{\text{ret}} := \text{NN}_{n_r}(q)$ as the set of n_r nearest neighbors of x from A. Note that only retrieving from the nearest neighbors might be counter-productive, since many nearly duplicate images could be retrieved and considered as negatives. This phenomenon gets sharpened if there is small/no distribution shift between adaptation data and the external pool of samples. $\mathrm{T}^3\mathrm{AR}$ solves this with a simple deduplication strategy applied to the retrieved data. We propose to randomly extract k samples among $\text{NN}_{r \times n_r}(q)$, *i.e.* we first select the top $r \cdot n_k$ retrieved data, and then uniformly sample a subset of k samples. In this way, even if the topmost retrieved samples are near duplicates to the query image the likelihood of treating them as negatives is reduced. In our experiments we find that $r = 5$ is a robust choice across different experiments.

To conclude, the set of negative examples we use in Eq. (1) is $\mathcal{N}_q = \mathcal{N}_q^{\text{lab}} \cup \mathcal{N}_q^{\text{ret}}$.

Ground truth labels vs pseudo-labels In $\mathrm{T}^3\mathrm{AR}$ it is possible to adapt pre-trained models not only at test-time (by the user) but also at train time (by the service provider) as new data become available. In the latter case, ground truth

labels might be available and should not be discarded. Our method can be modified to work with ground truth labels by incorporating them into its main objective in place of pseudo-labels so that ground truth labels are used to avoid same-class negatives in Eq. (1) and are used to directly supervise the model predictions. On the other hand, at test-time (when ground truth labels are not available), T^{3}AR exploits the close-set assumption and uses pseudo-labels [8, 34]. However, the quality of pseudo-labels is important, in Sec. 3.1.2 we propose a simple refinement strategy to get higher quality "filtered" ones.

3.1.2 Supervised/weakly-supervised objective

Since the contrastive loss does not update the linear classifier but only the features of the predictive model, we incorporate supervision into the objective function by exploiting labels y (if available) or, more generally, pseudo-labels $\bar{y}$, that are generated by the hypothesis $f_{A|T}$ [8, 34, 51]. However, pseudo-labels are known to be noisy, especially if T is different from S [8, 34], therefore we propose to further refine them by leveraging other augmented views of the same image x [65].

$$\bar{y}(x) = \arg\max_i \bar{f}_{A|T}(x) \tag{3}$$

where $\bar{f}_{A|T}(x)$ is obtained by averaging logits with respect to strong synthetic data augmentations of x. To improve efficiency of our method and reduce training time [8,34], we implement this exploiting a memory bank which contains past predicted logits and features (see Sec. 3.1.1).

We note that using "filtered" pseudo-labels to guide model adaptation can be interpreted as a form of consistency regularization or distillation, which, in the case of semi-supervised learning, has the main objective of propagating known labels towards unlabelled samples [34, 51]. Overall, our supervised loss/consistency regularization is implemented as:

$$\mathcal{L}_{\text{ce}}(x) = \mathbb{E}_{x \in \mathcal{D}_t} H(\bar{y}(x), f_{A|T}(t(x))) \tag{4}$$

where $H(a, b) = -\sum_{c=1}^{C} a_c \log b_c$ and $\bar{y}(x)$ is the "filtered" pseudo-label.

4. Experiments

4.1. Experimental setup

We evaluate T^{3}AR on standard train and test time adaptation benchmarks. At train-time, T^{3}AR is applied on fine-grained classification datasets as done in [1, 18]. In particular, we use MIT-67 [47], CUB-200 [56], FGVC-Aircraft [40], Stanford Cars [31], Stanford Dogs [29]. At test-time, following [8], we use a closed set benchmark composed of VisDA-C [46] and DomainNet-126 [45] (we use DomainNet-126 and not DomainNet since the latter has noisy labels [8]). DomainNet-126 contains 126 concepts shared across four domains (Real, Sketch, Clipart, Painting), while VisDA-C is a 12 class dataset that focuses on synthetic-to-real adaptation. To build the large pool of external data A we use images (without labels) from the following datasets: ImageNet1k [17], iNaturalist 2019 [25], Food-101 [5], Logo 2k+ [58], NWPU-RESISC 45 [14], iMaterialist Product [41]. Overall, size of the auxiliary dataset A is $\approx 2M$ images aggregated from a range of different domains and applications.

Table 1. **Comparison with transfer learning baselines.** Classification Top1 Accuracy (%) on fine-grained downstream datasets. Bold is the highest. Comparison of T^{3}AR with supervised transfer learning (fine-tuning) on ResNet50. We show that T^{3}AR performs on par with a strong supervised fine-tuning baseline on high shot fine-grained tasks. Moreover, when the number of samples allowed during adaptation is reduced (20% of the original datasets) we show that the use of an external data pool of images allows T^{3}AR to perform better on different fine-grained tasks.

	20% of samples				100% of samples			
	Sup.		Self Sup.		Sup.		Self Sup.	
Dataset	Sup. FT	T^{3}AR	Sup. FT	T^{3}AR	Sup. FT	T^{3}AR	Sup. FT	T^{3}AR
Stanford Cars	61.4	**66.0**	31.7	64.6	**93.5**	**93.5**	93.2	93.0
Aircrafts	11.8	35.0	39.9	**60.0**	86.4	88.4	88.2	**89.1**
CUB200	52.0	**55.5**	27.7	43.6	82.2	**82.4**	80.0	80.3
MIT-67	60.9	**67.6**	62.8	66.4	77.2	**77.6**	76.8	75.9
Stanford Dogs	86.8	**87.3**	40.9	56.5	**92.2**	89.6	76.5	81.9

Baselines For test-time adaptation we compare our method with both Unsupervised Domain Adaptation and Test-Time Adaptation methods. For UDA methods, we compare to CAN [28] and MCC [26] since they have been reported to be the best performing methods on our chosen benchmarks. For TTA we compare with TENT [57], SHOT [35] and AdaContrast [8]. We do not directly compare with TTT [53] since it requires to modify the pre-training objective function and is therefore not a truly test-time only adaptation method. All the baseline results on train-time training are obtained following supervised fine-tuning best practices (*e.g.* data augmentations such as MixUp [64] and RandAugment [15], linear warmup and cosine annealing learning rate schedules [19]) and running extensive hyper-parameter search (see Sec. C for details). Similarly to previous works [1, 8, 18], we use ConvNets architectures (ResNet50/101) pre-trained using both supervised [18] and self-supervised [7] objective functions on ImageNet1k.

4.2. Train time model adaptation with retrieval

At train-time T^{3}AR takes as input a model pre-trained on some pre-training data (either with supervision or self-supervision), a labelled dataset S and large database of images A and adapts the pre-trained features to the downstream task. Performance is evaluated on held out data T that is not used for further adaptation. This mimics the typi-

Table 2. **Comparison with UDA and TTA baselines.** Avg. Classification accuracy (%) on 7 domain shifts of DomainNet-126 and on 1 domain shifts of VisDA-C train $\rightarrow$ val for different target T dataset sizes (1%, 10% and 100%). Bold is the highest. T^{3}AR achieves the highest average performance when few samples are available for adaptation, 1% and 10% of the whole dataset.

Method	DomainNet-126			VisDA-C		
	1%	10%	100%	1%	10%	100%
CAN [28]	-	-	-	-	-	**87.2**
MCC [26]	-	-	48.9	-	-	78.8
Source only	55.6	55.6	55.6	43.8	43.8	43.8
TENT [57]	53.7	54.0	57.7	45.7	46.9	49.2
SHOT [35]	57.2	64.1	67.1	63.6	69.1	83.0
AdaContrast [8]	60.6	65.8	**67.8**	68.3	72.8	**87.2**
T^{3}AR	**63.5**	**66.3**	67.5	**70.2**	**77.5**	85.9

cal model customization scenario (transfer learning [1, 18]) solved with supervised fine-tuning. In this section, we pick S to be a labelled fine-grained classification dataset from the ones listed in Sec. 4.1.

In Tab. 1 we test how much retrieving samples from A help T^{3}AR at training time. We compare T^{3}AR adaptation against supervised fine-tuning of two different pre-trained backbone models both in the high and low data regime (see Sec. E for details on datasets subsampling). Therefore, we either use the whole downstream dataset (100%) or we subsample it by keeping only 20% of the labelled training data (all the remaining data are discarded, and no further used). Our models are pre-trained with a supervised objective or a self-supervised one on ImageNet1k. Note that T^{3}AR, compared to the baselines, improves feature adaptation in both data regimes and it is effective regardless of the backbone choice. In particular, supervised pre-trained features improve 13%/5% while self-supervised 30%/4% on the low and high data regime respectively. Our results show that A can be leveraged to add relevant information during adaptation even if the external data come from a different distribution. We further study the effect of adding more retrieved samples in Sec. 4.4, our results suggest that increasing the number of retrieved images saturates relatively early and the trade-off between computational cost (the more the retrievals the higher the training time) and performance is relatively stable across different datasets. In particular, the performance starts saturating as soon as the retrieved dataset is twice as large as the training dataset Fig. 8 in the appendix.

4.3. Test time model adaptation with retrieval

At test-time T^{3}AR takes as input a model pre-trained on the source dataset S whose labels space is the same as the one in the unlabelled target set T. However, the distribution of images in S need not be the same as in T (covariate shift). The performance of our method is evaluated on the average Top1 accuracy on different domains (7 for DomainNet-126 and 1 for VisDA-C). As in previous experiments, the auxiliary data pool A is taken as the concatenation of the datasets listed in Sec. 4.1. To compare our results with TTA literature [8, 35, 57], and only in this experiment, we fix the pre-trained backbones as the ones used in [8]. More specifically, we add a 256-dimensional bottleneck consisting of a fully-connected layer followed by a BatchNorm layer after the backbone, and apply WeightNorm on the classifier, for more details we refer to [8].

Table 3. **Do we need an expert retrieval module?** We compare train-time downstream Classification Top1 Accuracy (%) of T3AR on fine-grained classification tasks when equipped with random or expert retrieval module (*e.g.* CLIP, DINO). Even a non expert retrieval system does not jeopardize generalization. Nonetheless, the average relative performance drop w.r.t. to expert retrieval systems is $\approx 25\%$. And the stronger CLIP retrieval leads to better results.

Dataset	Random R	DINO	CLIP
Stanford Cars	61.4	62.4	**66.0**
Aircrafts	18.4	31.6	**35.0**
CUB200	48.2	54.0	**55.5**
MIT-67	62.2	66.6	**67.6**
Stanford Dogs	83.9	86.9	**87.3**

Previous results in the literature [8,35,53,57] assume that *all target data* T are used for adaptation. However, relying on plenty of samples for adaptation, even if unlabelled, could be a limiting factor in many real world scenarios. In Tab. 2 we test the capability of T3AR to efficiently adapt when little target data are available (1%, 10% and 100% of T). T^{3}AR achieves high Top1 average accuracy both on DomainNet-126 and VisDA-C benchmarks. In particular, the fewer the data available at test time the higher the performance gap w.r.t. other state of the art methods. As in the train-time experiment, we observe that the retrieval system plays an important role. In fact, while other methods [8, 35, 57] mainly rely on synthetic data augmentations to compensate for the lack of target data, our method also leverages retrieved real images that enable the learned features to better approximate the target data manifold.

4.4. Ablation studies

Do we need an expert retriever R? And does distribution shift of A w.r.t. T impact performance? Intuitively, the performance of T^{3}AR could be upper bounded by the Top-1 accuracy of its retrieval system. And the higher the domain gap of the retrieved samples from A w.r.t. target data in T the worse the downstream performance gets.

In Tab. 3 we answer the first question by comparing a random retrievals, and two expert retrieval systems, one based on DINO [7] and the other based on CLIP pre-training [48]. Both DINO and CLIP embeddings achieve high performance on zero-shot classification on the fine-

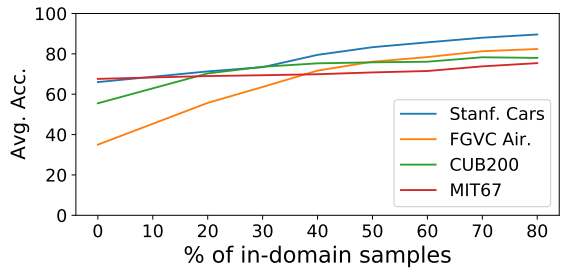

Figure 3. **How much distribution shift can T3AR tolerate?** We compare T^{3}AR in a train time setting on fine-grained classification datasets as the domain gap between the adaptation data and the auxiliary data A increases. To artificially control the distribution shift we progressively include more adaptation data, which are more likely to be retrieved, to the external pool. The higher the domain gap the lower the performance.

grained datasets we use (see Sec. B). In particular, in Tab. 3 we show that even a non-expert retrieval system does not completely jeopardize generalization, the average relative performance drop w.r.t. to an expert retrieval system is 20/25% and, for some datasets, it is comparable with the supervised fine-tuning results in Tab. 1. This observation suggests that even randomly retrieved images can act as a generic regularizer and do not harm generalization.

In Fig. 3 we answer the second question by artificially introducing a controlled distribution shift on the retrieved samples. In particular, we progressively include more data from the adaptation domain in the external pool that, in turn, are more likely to be retrieved by R. We note that the higher the domain gap is the lower the final accuracy gets, since finding hard (informative) negatives becomes harder.

Impact of the size of A. We test the sensitivity of T^{3}AR to the size of the external set by using a 10% subset of A, ImageNet-21k and its subset ImageNet-1k. In Tab. 4, we show that increasing the size of the external data pool leads to higher average accuracy. However, using larger datasets is not as helpful as having better domain coverage. The gaps are 0.4% (Subset-A $\rightarrow$ A), 0.7% (Subset-IN21k $\rightarrow$ IN21K), 1.2% (Subset-IN21k $\rightarrow$ A), 0.5% (IN21k $\rightarrow$ A). Interestingly, differently from [43] no task competition is present in T^{3}AR. In fact, thanks to the retrieval module that ignores what is not relevant, increasing the external dataset size strictly improves results.

Impact of the domain coverage of A. We replace our external dataset of 2M images with a 2M random subset of ImageNet-21k. In Tab. 4 we show that T^{3}AR still improves over the baselines of not using an external data pool (54.6% $\rightarrow$ 62.0% for train-time adaptation and 69.3% $\rightarrow$ 70.6% for test-time adaptation). However, our choice of A has better overall results (62.3% and 71.9% for train and test time respectively) due to better domain coverage.

Ablation over the composition of A. In Tab. 4 we ablate over the datasets used to build A. Removing ImageNet-1k from the external pool leads to 7.5 % average drop in performance, while dropping iNaturalist or Logo 2k is not as harmful and the average gap is $\approx$ 1%. In particular, we found that IN-1k (the largest dataset in A) provides most of the retrieved samples (more than 85%) both during Train- and Test-Time adaptation. However, there are some exceptions: CUB200 retrieves half of the data from iNaturalist, while DomainNet-126 (on all domains) retrieves more than 15% samples from Logo-2k and 5% from iNaturalist.

Table 4. **Ablations on the external dataset.** Accuracy on downstream tasks (rows) when using different external datasets (columns). Results are reported on the same 20% subsets used for train time experiments (see Tab. 1) and 10% subsets used for test time experiments (see Tab. 2). By A-IN1k, A-iNat, A-Logo we denote A ablated of the corresponding dataset.

		A		IN1k	IN21k		A-IN1k	A-iNat	A-Logo
	SOTA	100%	10%	100%	15%	100%			
Cars	61.4	66.0	65.8	65.6	66.8	**67.3**	60.3 (-5.7)	65.2 (-0.8)	65.2 (-0.8)
Air.	11.8	**35.0**	33.8	33.2	31.9	32.8	17.8 (-17.2)	34.3 (-0.7)	34.7 (-0.3)
CUB	52.0	**55.5**	55.1	53.7	53.7	54.5	53.4 (-2.1)	54.0 (-1.5)	55.4 (-0.1)
MIT	60.9	67.6	67.5	67.1	67.7	**68.5**	64.5 (-3.1)	67.4 (-0.2)	67.3 (-0.3)
Dogs	86.8	**87.3**	87.2	87.2	86.9	86.9	82.1 (-5.2)	86.8 (-0.5)	86.9 (-0.4)
Train Avg.	54.6	**62.3**	61.9	61.4	61.4	62.0	55.6 (-6.7)	61.5 (-0.8)	61.9 (-0.4)
DNet-126	65.8	**66.3**	65.7	63.8	64.5	64.7	57.6 (-8.7)	63.2 (-3.1)	62.8 (-3.5)
VisDA-C	72.8	**77.5**	77.0	76.6	75.1	76.5	66.8 (-10.7)	77.2 (-0.3)	76.8 (-0.7)
Test Avg.	69.3	**71.9**	71.4	70.2	69.8	70.6	62.2 (-9.7)	70.2 (-1.7)	69.8 (-2.1)
Avg.	58.8	**65.0**	64.6	63.9	63.8	64.5	57.5 (-7.5)	64.0 (-1.0)	64.2 (-0.8)

Sensitivity to the number of retrievals In Sec. F in the appendix we study the sensitivity of T^{3}AR to the number of allowed retrieved images. Our results across different datasets show a diminishing return in performance as the number of NNs increases (see Fig. 8). Since retrieving more samples increases (linearly) adaptation time, our experiments suggest that a trade-off, to discount compute over marginal accuracy improvements, is to retrieve no more than twice as many samples as the target dataset.

5. Conclusions

We introduced T^{3}AR to adapt pre-trained models both at train and test time by means of a retrieval module and a searchable pool of auxiliary samples. Differently from previously proposed methods [8, 65] that by-pass the lack of a adaptation data by introducing specific self-supervised objectives driven by data augmentations, T^{3}AR builds a self-supervised objective that is driven by real data, thus better capturing the target real data manifold. Furthermore, similarly to [8, 34], T^{3}AR exploits "filtered" pseudo-labels to align the output distribution of the model to the downstream class labels. T^{3}AR improves downstream fine-grained classification over standard fine-tuning baselines. Moreover, we compared our method against state of the art test-time adaptation algorithms [8, 35, 57, 65] and showed that it resulted in more robust and generalizable features, especially when the available data at test-time are scarce.

References

[1] Alessandro Achille, Aditya Golatkar, Avinash Ravichandran, Marzia Polito, and Stefano Soatto. Lqf: Linear quadratic fine-tuning, 2020. 1, 3, 6, 7, 13, 14

[2] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language model for few-shot learning, 2022. 1, 2

[3] Andreas Blattmann, Robin Rombach, Kaan Oktay, Jonas Müller, and Björn Ommer. Semi-parametric neural image synthesis, 2022. 2, 4

[4] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. Improving language models by retrieving from trillions of tokens, 2021. 2, 3, 4, 5

[5] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision*, 2014. 6, 14

[6] Pau Panareda Busto and Juergen Gall. Open set domain adaptation. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 754–763, 2017. 2

[7] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers, 2021. 5, 6, 7, 12, 13, 15

[8] Dian Chen, Dequan Wang, Trevor Darrell, and Sayna Ebrahimi. Contrastive test-time adaptation, 2022. 1, 2, 3, 4, 5, 6, 7, 8, 12, 13, 14, 15, 16

[9] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020. 5

[10] Ting Chen, Saurabh Saxena, Lala Li, David J. Fleet, and Geoffrey Hinton. Pix2seq: A language modeling framework for object detection, 2021. 1, 2

[11] Wenhu Chen, Hexiang Hu, Chitwan Saharia, and William W. Cohen. Re-imagen: Retrieval-augmented text-to-image generator, 2022. 2, 3

[12] Xiang Chen, Lei Li, Ningyu Zhang, Xiaozhuan Liang, Shumin Deng, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. Decoupling knowledge from memorization: Retrieval-augmented prompt learning, 2022. 3

[13] Yuhua Chen, Wen Li, Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Domain adaptive faster r-cnn for object detection in the wild, 2018. 3

[14] G. Cheng, J. Han, and X. Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017. 6, 14

[15] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical automated data augmentation with a reduced search space, 2019. 6

[16] Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021. 1

[17] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009. 6, 14

[18] Aditya Deshpande, Alessandro Achille, Avinash Ravichandran, Hao Li, Luca Zancato, Charless Fowlkes, Rahul Bhotika, Stefano Soatto, and Pietro Perona. A linearized framework and a new benchmark for model selection for fine-tuning, 2021. 1, 3, 5, 6, 7, 13, 14

[19] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2020. 6

[20] Xuefeng Du, Zhaoning Wang, Mu Cai, and Yixuan Li. Vos: Learning what you don't know by virtual outlier synthesis, 2022. 2

[21] Dong Gong, Lingqiao Liu, Vuong Le, Budhaditya Saha, Moussa Reda Mansour, Svetha Venkatesh, and Anton van den Hengel. Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection, 2019. 3

[22] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines, 2014. 3

[23] Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. Accelerating large-scale inference with anisotropic vector quantization. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3887–3896. PMLR, 13–18 Jul 2020. 5, 12, 16

[24] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning, 2019. 5, 12

[25] Grant Van Horn, Oisin Mac Aodha, Yang Song, Alexander Shepard, Hartwig Adam, Pietro Perona, and Serge J. Belongie. The inaturalist challenge 2017 dataset. *CoRR*, abs/1707.06642, 2017. 6, 14

[26] Ying Jin, Ximei Wang, Mingsheng Long, and Jianmin Wang. Minimum class confusion for versatile domain adaptation, 2019. 2, 3, 6, 7, 15, 16

[27] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019. 5, 12, 16

[28] Guoliang Kang, Lu Jiang, Yi Yang, and Alexander G Hauptmann. Contrastive adaptation network for unsupervised domain adaptation, 2019. 2, 3, 6, 7, 15, 16

[29] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO, June 2011. 6, 14

[30] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning, 2020. 5

[31] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013. 6, 14

[32] Dong-Hyun Lee. Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks. 2013. 2

[33] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc., 2020. 2, 3

[34] Junnan Li, Caiming Xiong, and Steven Hoi. Comatch: Semi-supervised learning with contrastive graph regularization, 2020. 2, 4, 6, 8, 12, 14

[35] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation, 2020. 1, 2, 3, 6, 7, 8, 12, 15

[36] Alexander Long, Wei Yin, Thalaiyasingam Ajanthan, Vu Nguyen, Pulak Purkait, Ravi Garg, Alan Blair, Chunhua Shen, and Anton van den Hengel. Retrieval augmented classification for long-tail visual recognition, 2022. 2, 3

[37] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. Learning transferable features with deep adaptation networks, 2015. 3

[38] Jiasen Lu, Christopher Clark, Rowan Zellers, Roozbeh Mottaghi, and Aniruddha Kembhavi. Unified-io: A unified model for vision, language, and multi-modal tasks, 2022. 1, 2

[39] Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner. Online continual learning in image classification: An empirical survey, 2021. 1

[40] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. Technical report, 2013. 6

[41] MalongTech. Imaterialist dataset, https://github.com/malongtech/imaterialist-product-2019. 6, 14

[42] Antoine Miech, Jean-Baptiste Alayrac, Ivan Laptev, Josef Sivic, and Andrew Zisserman. Thinking fast and slow: Efficient text-to-visual retrieval with transformers, 2021. 4

[43] Thao Nguyen, Gabriel Ilharco, Mitchell Wortsman, Sewoong Oh, and Ludwig Schmidt. Quality not quantity: On the interaction between dataset design and robustness of clip, 2023. 8

[44] Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Jie Ma, Alessandro Achille, Rishita Anubhai, Cicero Nogueira dos Santos, Bing Xiang, and Stefano Soatto. Structured prediction as translation between augmented natural languages. 2021. 2

[45] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation, 2018. 2, 3, 6, 13, 14

[46] Xingchao Peng, Ben Usman, Neela Kaushik, Judy Hoffman, Dequan Wang, and Kate Saenko. Visda: The visual domain adaptation challenge, 2017. 6, 13, 14

[47] Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 413–420, 2009. 6

[48] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. 2, 5, 7, 12

[49] Kuniaki Saito, Shohei Yamamoto, Yoshitaka Ushiku, and Tatsuya Harada. Open set domain adaptation by backpropagation, 2018. 2

[50] Burr Settles. Active learning literature survey. 2009. 2

[51] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence, 2020. 2, 4, 6, 14

[52] Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your vit? data, augmentation, and regularization in vision transformers. 2021. 12

[53] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts, 2019. 1, 2, 3, 6, 7

[54] Yonglong Tian, Olivier J. Henaff, and Aaron van den Oord. Divide and contrast: Self-supervised learning from uncurated data, 2021. 5, 15

[55] Yi-Hsuan Tsai, Wei-Chih Hung, Samuel Schulter, Kihyuk Sohn, Ming-Hsuan Yang, and Manmohan Chandraker. Learning to adapt structured output space for semantic segmentation, 2018. 3

[56] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 6, 14

[57] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization, 2020. 1, 2, 3, 6, 7, 8, 13, 15

[58] J. Wang, W. Min, S. Hou, S. Ma, Y. Zheng, H. Wang, and S. Jiang. Logo-2k+: A large-scale logo dataset for scalable logo classification. 2019.   . 6, 14

[59] Peng Wang, An Yang, Rui Men, Junyang Lin, Shuai Bai, Zhikang Li, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. Ofa: Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework, 2022. 1, 2

[60] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning, 2021. 1

[61] Xi Yan, David Acuna, and Sanja Fidler. Neural data server: A large-scale search engine for transfer learning data, 2020. 2, 4

[62] Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey, 2021. 2

[63] Dani Yogatama, Cyprien de Masson d'Autume, and Lingpeng Kong. Adaptive semiparametric language models, 2021. 2, 3

[64] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization, 2017. 6

[65] Marvin Zhang, Sergey Levine, and Chelsea Finn. Memo: Test time robustness via adaptation and augmentation, 2021. 2, 3, 6, 8, 13

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 $f_{A|T}$.

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

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

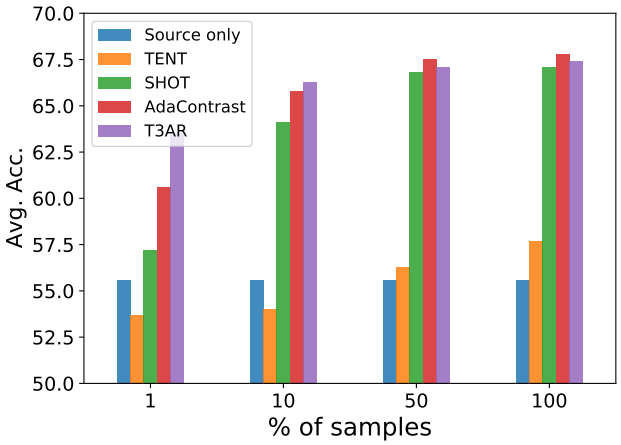

Figure 7. Classification Top1 Accuracy (%) of test-time adaptation methods on DomainNet-126 as the number of available target data T increases. We show that exploiting retrieved samples helps T^3AR especially in the low data regime.

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.

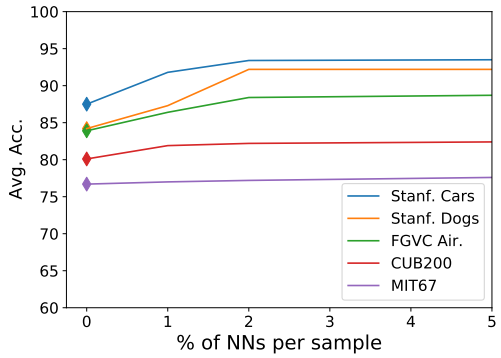

Figure 8. **Accuracy as a function of the number of the retrievals.** Classification Top1 Accuracy (%) of $\mathrm{T}^3\mathrm{AR}$ on train time adaptation on fine-grained classifications datasets as a function of the number of retrieved nearest neighbors. We denoted with diamonds the reference performance 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.