

Multi-Head Distillation for Continual Unsupervised Domain Adaptation in Semantic Segmentation

Antoine Saporta^{1,2} Arthur Douillard^{1,3} Tuan-Hung Vu² Patrick Pérez² Matthieu Cord^{1,2}

¹Sorbonne Université

²valeo.ai

³Heuritech

antoine.saporta@isir.upmc.fr, arthur.douillard@heuritech.com,

{tuan-hung.vu, patrick.perez, matthieu.cord}@valeo.com

Abstract

Unsupervised Domain Adaptation (UDA) is a transfer learning task which aims at training on an unlabeled target domain by leveraging a labeled source domain. Beyond the traditional scope of UDA with a single source domain and a single target domain, real-world perception systems face a variety of scenarios to handle, from varying lighting conditions to many cities around the world. In this context, UDAs with several domains increase the challenges with the addition of distribution shifts within the different target domains. This work focuses on a novel framework for learning UDA, continuous UDA, in which models operate on multiple target domains discovered sequentially, without access to previous target domains. We propose MuHDI, for Multi-Head Distillation, a method that solves the catastrophic forgetting problem, inherent in continual learning tasks. MuHDI performs distillation at multiple levels from the previous model as well as an auxiliary target-specialist segmentation head. We report both extensive ablation and experiments on challenging multi-target UDA semantic segmentation benchmarks to validate the proposed learning scheme and architecture.

1. Introduction

Autonomous vehicles have recently returned to the forefront with the dazzling progress of AI and deep learning as well as the emergence of many players around new forms of mobility. Thanks to the new generation of convolutional neural networks, cameras may be embedded to understand, in real-time, crucial aspects of the environment: nature and position of vehicles, pedestrians and stationary objects; position and meaning of lane markings, signs, traffic lights; drivable area; etc.

Today, these deep neural networks are trained in a fully-supervised fashion, requiring massive amounts of labeled data. While powerful, this form of training raises major is-

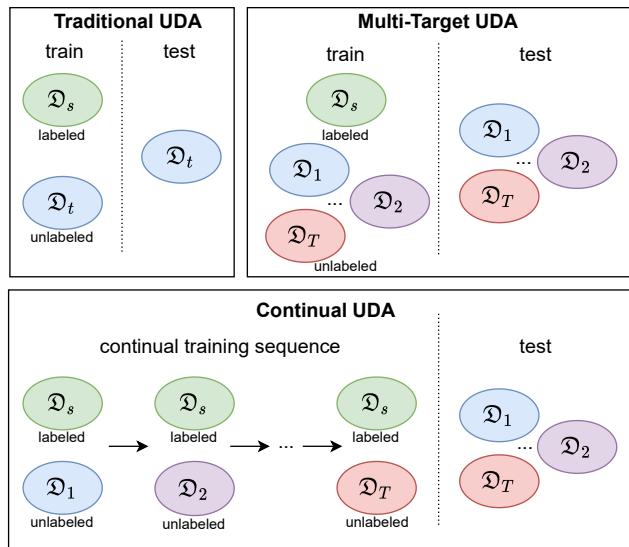


Figure 1. Traditional UDA trains a model on an unlabeled target domain \mathfrak{D}_t leveraging a labeled source domain \mathfrak{D}_s . Multi-target UDA trains a model on multiple unlabeled target domains $\mathfrak{D}_1, \mathfrak{D}_2, \dots, \mathfrak{D}_T$ simultaneously leveraging a labeled source domain \mathfrak{D}_s . Continual UDA is a more realistic and challenging setting in which the model sequentially learns unlabeled target domains $\mathfrak{D}_1, \mathfrak{D}_2, \dots, \mathfrak{D}_T$, without access of the previously seen target domains, by leveraging a labeled source domain \mathfrak{D}_s .

sues. Collecting large and diverse enough labeled datasets for supervised learning is a complex and expensive undertaking. Moreover, such datasets remain limited, considering the diversity, complexity, and unpredictability of environments a vehicle may encounter. A prime tool to tackle these issues is **Unsupervised Domain Adaptation (UDA)**. Its goal is to adapt to a set of unlabeled data, the *target domain*, sharing structures with another labeled dataset, the *source domain*, allowing supervised training though having some statistical distribution differences. Practically, one could, for example, use UDA to train a model on unlabeled

nighttime data as the target domain by taking advantage of labeled daytime data. Similarly, a promising practice is to use UDA to leverage synthetic data as the source domain, on which annotation is automatic and cheap, to train models on real-world data as the target domain.

While the standard UDA scenario, with one source domain and one target domain, is useful for autonomous driving applications, especially in the synthetic-to-real context, it is still very limited when considering practical use-cases. Indeed, autonomous vehicles may encounter a large variety of urban scene scenarios in the wild, such as varying weather conditions, lighting conditions, or different cities, each of those representing a specific domain. While standard UDA allows us to train a model on a particular unlabeled target domain, it may not be able to continuously improve and adapt to new target domains. Moreover, keeping access to previous real-world data is often unfeasible for various reasons, such as privacy, and a perception system must avoid forgetting about the domains it has previously seen when training on a new target domain.

In this work, we study a continual learning-inspired setting of UDA, which we call **Continual UDA**. The goal is to sequentially train a model on new target domains, one at a time, while maintaining its performance on the previous target domains without forgetting. [Figure 1](#) illustrates the differences between this novel setting and traditional and multi-target UDA. While multi-target UDA already extends traditional UDA to learn multiple unlabeled target domains at once, continual UDA is a more challenging setting in which the model must sequentially learn the new target domains, one domain at a time, and is evaluated on all the target domains, as in multi-target. The contributions of our work are threefold:

- We propose a new continual learning task for UDA with experimental benchmarks based on popular semantic segmentation datasets in the UDA community;
- To solve this new problem, we develop MuHDI, for Multi-Head Distillation, which explicitly solves the catastrophic forgetting problem by performing probability distribution distillation at multiple levels from the previous models as well as an auxiliary target-specialist segmentation head;
- We demonstrate the performance of the proposed approach compared to natural baselines as well as multi-target approaches, in which the target domains are all available simultaneously.

2. Related Works

Traditional Unsupervised Domain Adaptation. UDA has been an active research topic for a few years. Its objective is to learn on an unlabeled target domain \mathfrak{D}_t by

leveraging a labeled source domain \mathfrak{D}_s . The main challenge is to overcome the distribution shifts between the two domains. UDA generally relies either on aligning in some way the source and target features of the model to make them indistinguishable or on finding a transformation from source to target domain and using this transformation to train the model entirely in the target domain using the transformed source data. Different paradigms and strategies can be found in the literature to tackle the adaptation problem.

Direct distribution alignment approaches [17, 28, 29, 45, 49] directly minimize some distance or a measure of discrepancy between the two domains. Image-level approaches build transformations at the image level, either by directly translating the source image into the target domain before the neural network model dedicated to solving the task [2, 21, 53, 54] or by finding representations that allow reconstructing of images in either domain, independently of their domain of origin [3, 5, 33]. Finally, the last class of UDA approaches is based on adversarial training to align the distributions of some representations of source and target domains. Usually, the representations selected for adversarial alignment are either deep features extracted after the deep neural network backbone of the model right before the last layers dedicated to the task [15, 22, 48] or directly close-to-prediction representations at the output of the model [47, 52], which is the most popular and effective strategy today for semantic segmentation.

Multi-Domain Learning. Multi-domain learning is tackled in the literature under a variety of settings. Domain generalization [25, 31, 36, 56] is a challenging task close to the DA problem. Its objective is to learn a model from one or several training (source) domains that will be able to generalize to unseen testing (target) domains. While very close to a multi-domain UDA, domain generalization differs from it by the total absence of target data during training, leading to drastically different approaches to tackle these tasks.

Multi-source UDA [19, 35, 57] aims at training using an arbitrary number of source domains to learn better, more generalizable features for a single target domain. While one could train a model with a standard UDA approach on the combination of all the considered source domains, it proves to be inefficient due to the discrepancy between the domains. Multi-source approaches effectively account for these distribution shifts and leverage them to produce more generalizable features.

In Open-Compound Domain Adaptation (OCDA) [27, 37], the target domain may be considered as a combination of multiple homogeneous target domains – for instance, similar weather conditions such as ‘sunny’, ‘foggy’, etc. – where the domain labels are not known during training. Moreover, previously unseen target domains may be encountered during inference.

Multi-Target Unsupervised Domain Adaptation. Multi-target UDA is still a fairly recent setting in the literature and mostly tackles classification tasks. While multi-target UDA is close to the OCDA setting in that it considers multiple target domains instead of a single one, multi-target UDA differs from this last setting in that it assumes that the domain of origin is known during training and that no new domains are faced at test time. Thus, for instance, multi-target UDA would be better suited to DA to multiple cities than OCDA since the origin of the training data should be easily accessible. Two main scenarios emerge in the works on this task. In the first one, even though the target is considered composed of multiple domains with gaps and misalignments, the domain labels are unknown during training and test. [39] proposes an architecture that extracts domain-invariant features by performing source-target domain disentanglement. Moreover, it also removes class-irrelevant features by adding a class disentanglement loss. In a similar setting, [7] presents an adversarial meta-adaptation network that both aligns source with mixed-target features and uses an unsupervised meta-learner to partition the target inputs into clusters that are adversarially aligned. In the second scenario, the target identities are labeled on the training samples but remain unknown during inference. To handle it, [55] learns a common parameter dictionary from the different target domains and extracts the target model parameters by sparse representation; [16] adopts a disentanglement strategy by separately capturing both domain-specific private features and feature representations by learning a domain classifier and a class label predictor, and trains a shared decoder to reconstruct the input sample from those disentangled representations.

Tackling multi-target UDA in semantic segmentation has been proposed in two recent works. [23] trains multiple semantic segmentation models, each one expert on a specific domain. These domain-specific expert models collaborate by being trained on images from the other domains stylized in the domain of expertise while making sure that the predicted maps are coherent between the experts for a same original image. Finally, the knowledge of all these experts is transferred to another model, which serves as domain-generic student. [44] first proposes an approach combining two adversarial pipelines: the first one aims at discriminating the source domain to each individual target domain; the second one aims at discriminating each individual target domain to the other target domains. [44] then proposes a second strategy adopting a multi-teacher/single-student distillation approach to learn a segmentation model which is agnostic to the target domains.

Continual Learning. The task of continual learning aims at learning a constantly changing distribution. A naive mitigation

is to re-train the model from scratch on the updated dataset. However, it assumes that previous data is kept, which is often unfeasible for multiple reasons, including privacy. Thus, a continual model has to learn solely on the new data while remembering the previous data. As a result, the model faces the challenge of “catastrophic forgetting” [14, 42, 46] where the performance on previous samples drops. This problem can be mitigated by different approaches: rehearsal of a limited amount of previous data [18, 40] can reduce forgetting, but is memory-costly for high-resolution images required for semantic segmentation [12]. A second approach is to constrain the new model to be “similar” to the previous model. This similarity can be defined on the weights [24], the gradients [30], or even the probabilities [26] and the intermediary features [13]. More recently, continual models were adapted for semantic segmentation [4, 11, 32] with success, but they restricted themselves to supervised tasks on a single dataset and not unsupervised adaptation across multiple domains.

Closer to UDA, [51] proposes a continual learning-inspired setting to the domain adaptation problem: a model is learned on multiple labeled domains seen sequentially which are all considered for evaluation. This setting, being fully supervised, substantially differs from the novel setting proposed in our work, which is unsupervised for all the target domains and only leverages a labeled source domain. [43] studies a UDA setting closer to ours. The model is sequentially trained on unlabeled target domains but requires storage of representative samples of the previous domains which are used for rehearsal. Furthermore, [43, 51] consider classification tasks while the focus of our work is semantic segmentation.

3. Continual UDA Setting

We define in this section the continual unsupervised domain adaptation problem, inspired by continual learning tasks, and we propose extensions of traditional unsupervised domain adaptation methods as baselines for this new task. This problem is illustrated in Figure 1.

Problem Setting. Let us consider $T \geq 2$ distinct target domains $\mathfrak{D}_t, t \in [T] = \{i \in \mathbb{N} \mid 1 \leq i \leq T\}$, to be jointly handled by the model. They are represented by unlabeled training sets $\mathcal{X}_t, t \in [T]$. As in traditional UDA settings, we assume that the labeled training examples $(\mathbf{x}, \mathbf{y}) \in \mathcal{X}_s \times \mathcal{Y}_s$ stem from a single source domain, a specific synthetic environment for instance.

This setting differs from the multi-target UDA setting: we assume here that the different target domains can only be accessed sequentially and one at a time during training. More precisely, during step $t \in [T]$ of the training, the model only has access to a single target dataset, \mathcal{X}_t , and cannot access ever again the target datasets $\mathcal{X}_k, k \in [t-1]$, it

has previously learned. Nevertheless, we consider that the source domain dataset $(\mathcal{X}_s, \mathcal{Y}_s)$, which is often synthetic, is always accessible during training. We denote $F_{(1:t)}$ the continual model that one obtains after sequentially training from target domains 1 to t .

The objective of continual DA is the same as in a multi-target setting: train a single semantic segmentation model, $F_{(1:T)}$, that achieves equally good results on all target-domain test sets. Also, while the target domain of origin is known for all unlabeled training examples, we assume, as in multi-target approaches of [16, 23, 44, 55], that this information is not accessible at test time.

The continual DA setting brings a new challenge compared to the multi-target one: the model must not forget the previous target domains it has learned before (*rigidity*) while still being able to adapt to new domains (*plasticity*).

Revisiting Adversarial UDA. We adapt the training procedure of state-of-the-art UDA approaches like [47] or [52] to this new setting. In continual UDA, the model always has access during training to a single source domain and a single target domain. Traditional UDA approaches can easily be adapted to this setting: at each iteration $t \in [T]$, the model is trained on the source dataset \mathcal{X}_s and the current target dataset \mathcal{X}_t , initialized from the model trained at the previous iteration. This way, the model is trained in a UDA fashion sequentially on all the target domains. However, this “continual baseline” does not directly tackle catastrophic forgetting of old domains. The following section describes our strategy to explicitly prevent catastrophic forgetting.

4. Multi-Head Distillation Framework

This section presents our approach to continual UDA for semantic segmentation: MuHDI, for Multi-Head Distillation. Given T target domains, training is done in T sequential steps. To ease explanation, we decompose the continual model $F_{(1:t)}$ trained at step t into the feature extractor $F_{(1:t)}^{\text{feat}}$ and the pixel-wise classifier $F_{(1:t)}^{\text{cls}}$. We denote $F_{(1:t)} = [F_{(1:t)}^{\text{feat}}, F_{(1:t)}^{\text{cls}}]$ with corresponding parameters $\theta_{(1:t)} = [\theta_{(1:t)}^{\text{feat}}, \theta_{(1:t)}^{\text{cls}}]$.

4.1. First step $t = 1$

In the first iteration, the segmentation model $F_{(1:1)} = F_1 = [F_{(1:1)}^{\text{feat}}, F_1^{\text{cls}}]$ ¹ is trained with the traditional UDA approach AdvEnt [52] on the single-target problem with the source domain \mathcal{D}_s and the target domain \mathcal{D}_1 . A discriminator D_1 is trained on the source dataset \mathcal{X}_s and target dataset

\mathcal{X}_1 by minimizing the classification loss (taking $t = 1$):

$$\begin{aligned} \mathcal{L}_{D_t} = & \frac{1}{|\mathcal{X}_s|} \sum_{\mathbf{x}_s \in \mathcal{X}_s} \mathcal{L}_{\text{BCE}}(D_t(\mathbf{I}_{\mathbf{x}_s}), 1) \\ & + \frac{1}{|\mathcal{X}_t|} \sum_{\mathbf{x}_t \in \mathcal{X}_t} \mathcal{L}_{\text{BCE}}(D_t(\mathbf{I}_{\mathbf{x}_t}), 0), \end{aligned} \quad (1)$$

where \mathcal{L}_{BCE} is the Binary Cross-Entropy loss and $\mathbf{I}_{\mathbf{x}}$ is the weighted self-information map derived from the output $\mathbf{P}_{\mathbf{x}}$ of the model as:

$$\mathbf{I}_{\mathbf{x}} = -\mathbf{P}_{\mathbf{x}} \log \mathbf{P}_{\mathbf{x}}, \quad (2)$$

with pixelwise multiplication and logarithm. Concurrently, the semantic segmentation model $F_1 = [F_{(1:1)}^{\text{feat}}, F_1^{\text{cls}}]$ is trained over its parameters $\theta_1 = [\theta_{(1:1)}^{\text{feat}}, \theta_1^{\text{cls}}]$ not only to minimize the supervised segmentation loss $\mathcal{L}_{F_1, \text{seg}}$ on source-domain data, but also to fool the discriminator D_1 via minimizing an adversarial loss $\mathcal{L}_{F_1, \text{adv}}$. The final objective reads (taking $t = 1$):

$$\begin{aligned} \mathcal{L}_{F_t} = & \underbrace{\frac{1}{|\mathcal{X}_s|} \sum_{\mathbf{x}_s \in \mathcal{X}_s} \mathcal{L}_{\text{seg}}(\mathbf{P}_{\mathbf{x}_s}, \mathbf{y}_s)}_{\mathcal{L}_{F_t, \text{seg}}} \\ & + \lambda_{\text{adv}} \underbrace{\frac{1}{|\mathcal{X}_t|} \sum_{\mathbf{x}_t \in \mathcal{X}_t} \mathcal{L}_{\text{BCE}}(D_t(\mathbf{I}_{\mathbf{x}_t}), 1)}_{\mathcal{L}_{F_t, \text{adv}}} \end{aligned} \quad (3)$$

with a weight λ_{adv} balancing the two terms; \mathcal{L}_{seg} is the pixel-wise cross-entropy loss. During training, one alternately minimizes the two losses \mathcal{L}_{D_1} and \mathcal{L}_{F_1} .

4.2. Subsequent steps $t \geq 2$

At iteration $t \geq 2$, the segmentation model $F_{(1:t)}$ is trained on the source domain \mathcal{D}_s and the target domain \mathcal{D}_t to run on the target domains 1 to t . We here re-design the classification part of $F_{(1:t)}$ with two pixel-wise classifiers, referred to as *target-specialist* F_t^{cls} and *target-generalist* $F_{(1:t)}^{\text{cls}}$. At test time, only the target-generalist $F_{(1:t)}^{\text{cls}}$ is used for prediction. Note that in the special case $t = 1$ the target-generalist $F_{(1:1)}^{\text{cls}}$ is equivalent to the target-specialist F_1^{cls} due to dealing with a single target domain. Figure 2 provides an overview of the training scheme of $F_{(1:t)}$ when $t \geq 2$.

Target-specialist head. In this step t , the network has a *target t -specialist* instrumental segmentation head F_t^{cls} based on the feature extractor $F_{(1:t)}^{\text{feat}}$. This classifier handles the specific domain shift between source and target t by performing output-space adversarial alignment between the two domains. This classification head is associated with a domain discriminator D_t to classify source vs. target t .

¹See Sec. 4.2 for further explanations of this modified notation.

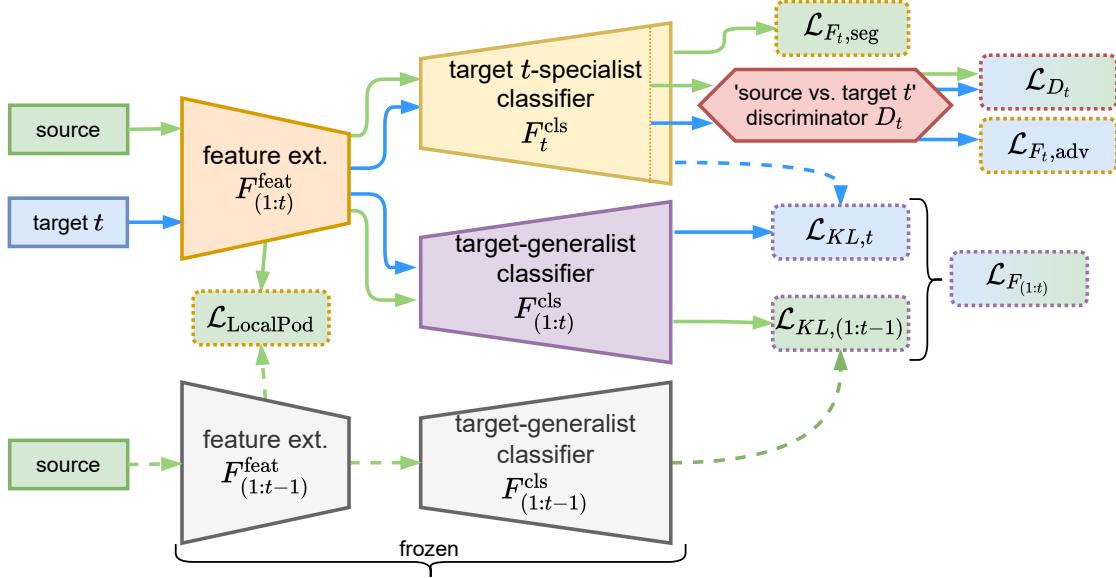


Figure 2. **MuHDI approach to continual UDA.** When discovering a new target domain t , MuHDI learns its target-generalist model $F_{(1:t)} = [F_{(1:t)}^{\text{feat}}, F_{(1:t)}^{\text{cls}}]$ using knowledge distillation from both a target t -specialist segmentation head F_t^{cls} trained adversarially for this target domain, and the frozen segmentation model from the previous training step $F_{(1:t-1)} = [F_{(1:t-1)}^{\text{feat}}, F_{(1:t-1)}^{\text{cls}}]$. In combination with this architectural design, the training losses are indicated and further developed in the text. In particular, the Local POD loss $\mathcal{L}_{\text{LocalPod}}$ is introduced to further prevent catastrophic forgetting. The losses are not back-propagated into dotted arrows.

The training objectives are similar to those used in single-target models as described for the first iteration (Sec. 4.1). To train the discriminator D_t on the source dataset \mathcal{X}_s and target dataset \mathcal{X}_t , one minimizes the classification loss as defined in [Equation 1](#).

Concurrently, the target t -specialist semantic segmentation model $F_t = [F_{(1:t)}^{\text{feat}}, F_t^{\text{cls}}]$ is trained over its parameters $\theta_t = [\theta_{(1:t)}^{\text{feat}}, \theta_t^{\text{cls}}]$ not only to minimize the supervised segmentation loss $\mathcal{L}_{F_t, \text{seg}}$ on source-domain data, but also to fool the discriminator D_t via minimizing an adversarial loss $\mathcal{L}_{F_t, \text{adv}}$. The final objective \mathcal{L}_{F_t} reads as in [Equation 3](#). During training, one alternately minimizes the two losses \mathcal{L}_{D_t} and \mathcal{L}_{F_t} .

Target-generalist head. The *target-generalist* segmentation head $F_{(1:t)}^{\text{cls}}$, which is eventually kept as classification head of the model, is trained to perform well on all the target domains from 1 to t . The knowledge from the target t -specialist branch is distilled to the target-generalist branch via a teacher-student strategy by minimizing the Kullback-Leibler (KL) divergence between the predictions of the two segmentation heads on the target domain t . For a given sample $\mathbf{x}_t \in \mathcal{X}_t$, we compute the KL loss

$$\mathcal{L}_{\text{KL},t}(\mathbf{x}_t) = \sum_{h=1}^H \sum_{w=1}^W \sum_{c=1}^C \mathbf{P}_{t,\mathbf{x}_t}[h,w,c] \log \frac{\mathbf{P}_{t,\mathbf{x}_t}[h,w,c]}{\mathbf{P}_{(1:t),\mathbf{x}_t}[h,w,c]}, \quad (4)$$

where $H \times W$ is image dimension; $\mathbf{P}_{t,\mathbf{x}_t}$ and $\mathbf{P}_{(1:t),\mathbf{x}_t}$ are soft-segmentation predictions coming from the target-specialist F_t^{cls} and the target-generalist $F_{(1:t)}^{\text{cls}}$ respectively.

4.3. Preventing catastrophic forgetting

Furthermore, the model must not forget what it has learned in the previous training iterations about the other target domains from 1 to $t-1$. Without proper constraint, the model may be subject to catastrophic forgetting, degrading its performance on previous target domains. Thus, we consider additional losses based on the model’s previous iteration when training the model on the new target domain. The aim of these losses is to make sure that the new model keeps similar features to its previous iteration in order to keep similar results on the previous target domains, on which the old model was performing well.

Ideally, one would want to use images from the previous target domains to get the features and results of the old model and make sure they are as close as possible to those of the currently-trained model. However, in our continual setting, one cannot access images of the previous domains anymore. Thanks to adversarial alignment, we here assume that the features on the source domain are close enough to the features of the previous target domains to use them as proxy. Under this assumption, we constrain source domain features of the current model with source domain features of the previous model.

Practically, we perform a similar distribution distillation

with KL distillation between the previously-trained segmentation model $F_{(1:t-1)}$, supposed to perform well on targets 1 to $t-1$, and the currently-trained segmentation model $F_{(1:t)} = [F_{(1:t)}^{\text{feat}}, F_{(1:t)}^{\text{cls}}]$ we want to handle all targets 1 to t . Without access to images from domains 1 to $t-1$, we use source images as proxy. For a given source sample $\mathbf{x}_s \in \mathcal{X}_s$, we compute the KL loss

$$\mathcal{L}_{\text{KL},(1:t-1)}(\mathbf{x}_s) = \sum_{h=1}^H \sum_{w=1}^W \sum_{c=1}^C \mathbf{P}_{(1:t-1),\mathbf{x}_s}[h,w,c] \log \frac{\mathbf{P}_{(1:t-1),\mathbf{x}_s}[h,w,c]}{\mathbf{P}_{(1:t),\mathbf{x}_s}[h,w,c]}, \quad (5)$$

where $\mathbf{P}_{(1:t-1),\mathbf{x}_s}$ and $\mathbf{P}_{(1:t),\mathbf{x}_s}$ are soft-segmentation predictions coming from the previous iteration of the target-generalist classifier $F_{(1:t-1)}^{\text{cls}}$, previously trained for targets 1 to $t-1$ and now frozen, and the currently-trained target-generalist head $F_{(1:t)}^{\text{cls}}$, respectively.

With the addition of this loss, the minimization objective of the target-generalist model $F_{(1:t)} = [F_{(1:t)}^{\text{feat}}, F_{(1:t)}^{\text{cls}}]$ over its parameters $\theta_{(1:t)} = [\theta_{(1:t)}^{\text{feat}}, \theta_{(1:t)}^{\text{cls}}]$, or distribution distillation loss, then reads:

$$\begin{aligned} \mathcal{L}_{F_{(1:t)}} &= \frac{1}{|\mathcal{X}_t|} \sum_{\mathbf{x}_t \in \mathcal{X}_t} \mathcal{L}_{\text{KL},t}(\mathbf{x}_t) \\ &+ \lambda_{\text{prev}} \frac{1}{|\mathcal{X}_s|} \sum_{\mathbf{x}_s \in \mathcal{X}_s} \mathcal{L}_{\text{KL},(1:t-1)}(\mathbf{x}_s), \end{aligned} \quad (6)$$

with weight λ_{prev} to balance the distribution distillation from the previous model.

Along with this distribution distillation loss, we also want to specifically enforce the current feature extractor F^{feat} to produce features at each layer close to the previous iteration of the feature extractor. This kind of approach helps alleviate the catastrophic forgetting in continual learning problems. Practically, we propose to use the Local POD distillation $\mathcal{L}_{\text{LocalPod}}(\theta^{\text{feat}})$ proposed in [11]. Denoting $\psi(\mathbf{u})$ the concatenation of width-pooled slices and height-pooled slices over multiple scales of \mathbf{u} , the Local POD distillation loss is defined, on each sample \mathbf{x}_s of the source dataset \mathcal{X}_s , and over each layer activation $F_{(1:t)}^{\text{feat}(l)}(\mathbf{x}_s)$ and $F_{(1:t-1)}^{\text{feat}(l)}(\mathbf{x}_s)$, $l \in [L]$, of the the feature extractors $F_{(1:t)}^{\text{feat}}$ and $F_{(1:t-1)}^{\text{feat}}$, as:

$$\mathcal{L}_{\text{LocalPod}}(\theta^{\text{feat}}) = \frac{1}{L} \sum_{l=1}^L \left\| \psi(F_{(1:t)}^{\text{feat}(l)}) - \psi(F_{(1:t-1)}^{\text{feat}(l)}) \right\|_F^2. \quad (7)$$

It is a multi-scale pooling distillation method that aims to preserve spatial relationships on the intermediate features. A more in-depth definition of the loss is developed in [11].

Note that this distillation loss was originally proposed in [11] in a continual semantic segmentation setting where new classes were added at each iteration while staying in the same domain. The setting considered here is notably

different since the classes do not change throughout training, but the model encounters new target domains at each iteration. Furthermore, the target domains are not labeled, while segmentation maps are available for all images during training in the setting of [11].

Overall, the minimization objective of the semantic segmentation model F over θ can be written as:

$$\mathcal{L}_F = \mathcal{L}_{F_t, \text{seg}} + \lambda_{\text{adv}} \mathcal{L}_{F_t, \text{adv}} + \lambda_{\text{dd}} \mathcal{L}_{F_{(1:t)}} + \lambda_{\text{fd}} \mathcal{L}_{\text{LocalPod}}, \quad (8)$$

with factors $\lambda_{\text{adv}}, \lambda_{\text{dd}}, \lambda_{\text{fd}}$ balancing adversarial training, distribution distillation and feature distillation, respectively.

5. Experimental Results

5.1. Experimental Setting

Continual Protocol. Each continual UDA experiment follows the same training protocol: when training a model on a continual benchmark $\mathcal{D}_s \rightarrow \mathcal{D}_1 \rightarrow \mathcal{D}_2 \rightarrow \dots \rightarrow \mathcal{D}_T$, the model trains in T steps. At each step t , the model is initialized from the previous model at iteration $t-1$ and is trained in a UDA fashion on $\mathcal{D}_s \rightarrow \mathcal{D}_t$, i.e. on the labeled source domain \mathcal{D}_s and the unlabeled target domain \mathcal{D}_t . Note that the model leverages at each step the supervision from the same source domain and that it cannot train again on the target domains it has previously seen.

Datasets. We build our experiments on four popular semantic segmentation datasets. GTA5 [41] is a dataset of 24,966 synthetic images of size 1920×1080 rendered using the eponymous open-world video game. We consider GTA5 as source domain and its training dataset is always available. Cityscapes [8] contains 2048×1024 -sized labeled urban scene images from cities around Germany, split in training and validation sets of 2,975 and 500 samples respectively. Mapillary Vistas [34], Mapillary in short, is a dataset collected in multiple cities around the world, which is composed of 18,000 training and 2,000 validation labeled scenes of varying sizes. IDD [50] is an Indian urban dataset having 6,993 training and 981 validation labeled scenes of size 1280×720 .

As in [44], we standardize the label set with the 7 super classes common to all four datasets. In this continual unsupervised domain adaptation settings, we train the model sequentially on the target datasets, one after the other. Nonetheless, evaluation of continual models is performed on all the seen target datasets.

Implementation Details. The experiments are conducted with PyTorch [38]. The adversarial framework is based on AdvEnt’s published code² [52]. The semantic segmentation model is DeepLab-V2 [6], built upon the ResNet-

²<https://github.com/valeoai/ADVENT>

GTA5 → Cityscapes → IDD					
Setting	Method	Target	mIoU	mIoU Avg.	
Multi-Target Oracle	Multi-Target Baseline [52]	Cityscapes	70.0	67.4	
	Multi-Dis. [44]	Cityscapes	68.9		
	MTKT [44]	Cityscapes	70.4	68.2	
Continual	Continual Baseline [52]	Cityscapes	63.6	64.5	
	MuHDI (ours)	Cityscapes	68.0	66.7	
		IDD	65.4		
		IDD	65.3		

Table 1. **Continual UDA segmentation performance on GTA5 → Cityscapes → IDD (two steps).** ‘Setting’ indicates if the experiment is multi-target (oracle, simultaneous training on all the target domains) or continual (target domains discovered sequentially, two-step training). **Bold** indicates the best continual performance in terms of mIoU.

101 [20] backbone first initialized with ImageNet [9] pre-trained weights. In a continual setting, when considering a new target domain, the model is initialized with the weights of the previously-trained model. All semantic segmentation models are trained by SGD [1] with learning rate $2.5 \cdot 10^{-4}$, momentum 0.9 and weight decay 10^{-4} . We train the discriminators using an Adam optimizer [10] with learning rate 10^{-4} . All experiments are conducted at the 640×320 resolution. For MuHDI, the weights λ_{prev} and λ_{dist} , balancing the distribution distillation from the previously-trained model, are set to 10^{-5} in all experiments. We use the authors’ implementation of the Local POD loss ³.

Evaluation. Semantic segmentation models are generally evaluated in terms of Intersection over Union (IoU), also known as Jaccard index, per class or mean Intersection over Union (mIoU) over all classes, expressed in percentage. In our specific scenario where models are evaluated on multiple target domains, we also include the mIoU averaged over the target domains (mIoU Avg.) as in [44].

5.2. Two-Target Domain Results on GTA5 → Cityscapes → IDD

We first consider the two-step continual task GTA5 → Cityscapes → IDD: the models are first trained on GTA5 → Cityscapes, then on GTA5 → IDD. Table 1 shows the results on this benchmark. First, the results of the continual baseline highlight the catastrophic forgetting problem. Indeed, while the model on the previous step (GTA5 → Cityscapes)

³https://github.com/arthurdouillard/CVPR2021_PLOP

Training Step	Distribution Distillation	Feature Distillation	Multi Heads		
			mIoU Cityscapes	mIoU IDD	mIoU Avg.
$t = 1$ G → C			69.0	61.5	65.2
$t = 2$ G → C → I	✓		63.6	65.4	64.5
		✓	65.3	65.3	65.3
	✓		66.4	65.5	65.9
	✓	✓	66.8	65.5	66.2
	✓	✓	68.0	65.3	66.7

Table 2. **Ablation study of the MuHDI architecture for continual UDA.** All the models are trained in a similar fashion in the first training step ($t = 1$) on GTA5 → Cityscapes (‘G → C’) with AdvEnt [52]. In the second step ($t = 2$), the models are trained on GTA5 → IDD from the previous iteration (‘G → C → I’) and are decomposed into multiple blocks .

has a 69.0% mIoU score on Cityscapes, the performance on Cityscapes of the continual baseline drops to 63.6% mIoU (-5.4%). MuHDI, which directly tackles catastrophic forgetting in its design, only loses 1.0% mIoU on Cityscapes from the previous step while having similar performance on the new target domain IDD compared to the continual baseline. Moreover, the performance of MuHDI in terms of mIoU Avg. gets close to the multi-target models, which serve as oracles, further demonstrating its efficiency.

Ablation Study. We perform an ablation study on this setting of the proposed method MuHDI after this second training step. The results are displayed in Table 2.

In this table, we analyze the impact of the multiple elements of MuHDI: probability distribution distillation from the previous model (‘Distribution Distillation’), feature distillation from the previous model with $\mathcal{L}_{\text{LocalPod}}$ (‘Feature Distillation’), and the decomposition into target-specialist and target-generalist branches (‘Multi Heads’).

First, we note that all the continual experiments exhibit comparable performance on IDD, on which they were trained last: from 65.2% to 65.5% mIoU on IDD, much higher than the 61.5% mIoU on IDD of the previous model, not trained on this domain.

Then, we note that the continual baseline (second row of Table 2 with 65.5% mIoU average), not implementing any of these elements, suffers catastrophic forgetting on the previous Cityscapes target domain, dropping from 69.0% to 63.6% mIoU. Despite the improvement on IDD, on which it was trained last, the overall performance is lower than the previous model (difference of -0.7% mIoU Avg.).

Implemented on their own, both the distribution distillation from the previous model and the feature distillation with LocalPod help limit the catastrophic forgetting on Cityscapes. Furthermore, they prove to be complementary, exhibiting better results on Cityscapes when combined.

GTA5 → Cityscapes → IDD → Mapillary												
Setting	Method	Target	flat	constr.	object	nature	sky	human	vehicle	mIoU	mIoU Avg.	
Multi-Target Oracle	Multi-Target Baseline [52]	Cityscapes	93.6	80.6	26.4	78.1	81.5	51.9	76.4	69.8	67.8	
		IDD	92.0	54.6	15.7	77.2	90.5	50.8	78.6	65.6		
		Mapillary	89.2	72.4	32.4	73.0	92.7	41.6	74.9	68.0		
	Multi-Dis. [44]	Cityscapes	94.6	80.0	20.6	79.3	84.1	44.6	78.2	68.8	68.2	
		IDD	91.6	54.2	13.1	78.4	93.1	49.6	80.3	65.8		
		Mapillary	89.0	72.5	29.3	75.5	94.7	50.3	78.9	70.0		
	MTKT [44]	Cityscapes	94.6	80.7	23.8	79.0	84.5	51.0	79.2	70.4	69.1	
		IDD	91.7	55.6	14.5	78.0	92.6	49.8	79.4	65.9		
		Mapillary	90.5	73.7	32.5	75.5	94.3	51.2	80.2	71.1		
Continual	Continual Baseline [52]	Cityscapes	92.9	79.0	18.7	76.9	84.1	47.3	72.9	67.4	67.0	
		IDD	91.8	51.1	11.6	79.0	91.6	47.5	72.5	63.6		
		Mapillary	90.3	71.7	30.1	76.1	93.9	50.2	77.3	70.0		
	MuHDI (ours)	Cityscapes	94.9	80.2	19.3	79.4	80.7	53.2	78.2	69.4	68.2	
		IDD	92.5	54.1	12.0	79.2	92.7	48.0	76.6	65.0		
		Mapillary	91.0	73.2	29.2	76.0	94.1	50.0	78.0	70.2		

Table 3. **Continual UDA segmentation performance on GTA5 → Cityscapes → IDD → Mapillary (three steps).** ‘Setting’ indicates if the experiment is multi-target (oracle, simultaneous training on all the target domains) or continual (target domains discovered sequentially, three-step training). **Bold** indicates the best continual performance in terms of mIoU.

Finally, MuHDI adds the decomposition into target-specific and target-agnostic branches, which lessens catastrophic forgetting even further. MuHDI performance on Cityscapes is only 1.0% mIoU lower than the previous model trained only on Cityscapes while being competitive on IDD. Overall, the performance of MuHDI in terms of mIoU Avg. is significantly higher than the other models, notably +2.2% greater than the continual baseline.

5.3. Three-Target Domain Results on GTA5 → Cityscapes → IDD → Mapillary

We consider the challenging setup involving three target domains – Cityscapes, Mapillary and IDD – discovered sequentially in a three-step continual learning setting and show results in Table 3. We compare results of continual UDA with multi-target UDA from [44]. Due to the simultaneous availability of data for all the target domains, the multi-target setting is easier than the continual setting and the performance of multi-target experiments are expected to be higher than those of continual UDA experiments.

The continual baseline performs worse than all the multi-target models with at least a –0.8% mIoU Avg. decrease. In particular, the results on the Cityscapes and IDD datasets, which have been seen in early continual training steps, are significantly lower than those of all the other models due to catastrophic forgetting. Moreover, its performance is notably degraded on the *human* and *vehicle* classes compared

to the better performing multi-target models, which is especially critical for autonomous driving applications.

On the other hand, MuHDI exhibits comparable performance to the rather competitive Multi-Dis. multi-target model [44] with a 68.2% mIoU Avg., proving the efficiency of the proposed continual learning framework.

6. Conclusion

Practical applications of UDA require models that perform on a multitude of different domains, such as multiple cities or various weather conditions. While effective in the traditional single-target setting, standard UDA strategies do not easily improve and learn new target domains when confronted to new environments.

We proposed MuHDI, for Multi-Head Distillation, to tackle the continual UDA problem. MuHDI adapts to each new target domain using a probability distribution distillation strategy from a target-specialist head to a target-generalist head, while also distilling the distribution and features from the previous model, trained to perform on all the previous target domains. The proposed benchmarks and architecture deliver competitive baselines for future developments of real-world UDA scenarios like continual UDA.

Acknowledgments: This work was supported by ANR grand VISA DEEP (ANR-20-CHIA-0022), and HPC resources of IDRIS under the allocation 2021-[AD011012153R1] made by GENCI.

References

[1] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proc. Int. Conf. Computational Statistics*. 2010. 7

[2] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. *Proc. Conf. Comp. Vision Pattern Rec.*, 2017. 2

[3] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks. *Int. Conf. Neural Inf. Proc. Systems*, 2016. 2

[4] Fabio Cermelli, Massimiliano Mancini, Samuel Rota Bulò, Elisa Ricci, and Barbara Caputo. Modeling the background for incremental learning in semantic segmentation. In *Proc. Conf. Comp. Vision Pattern Rec.*, 2020. 3

[5] Wei-Lun Chang, Hui-Po Wang, Wen-Hsiao Pengg, and Wei-Chen Chiu. All about structure: Adapting structural information across domains for boosting semantic segmentation. In *Proc. Conf. Comp. Vision Pattern Rec.*, 2019. 2

[6] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan Yuille. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Trans. Pattern Anal. Machine Intell.*, 2018. 6

[7] Ziliang Chen, Jingyu Zhuang, Xiaodan Liang, and Liang Lin. Blending-target domain adaptation by adversarial meta-adaptation networks. In *Proc. Conf. Comp. Vision Pattern Rec.*, 2019. 3

[8] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. Conf. Comp. Vision Pattern Rec.*, 2016. 6

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proc. Conf. Comp. Vision Pattern Rec.*, 2009. 7

[10] Jimmy Ba Diederik P. Kingma. Adam: A method for stochastic optimization. In *Proc. Int. Conf. Learning Representations*, 2015. 7

[11] Arthur Douillard, Yifu Chen, Arnaud Dapogny, and Matthieu Cord. Plop: Learning without forgetting for continual semantic segmentation. In *Proc. Conf. Comp. Vision Pattern Rec.*, 2021. 3, 6

[12] Arthur Douillard, Yifu Chen, Arnaud Dapogny, and Matthieu Cord. Tackling catastrophic forgetting and background shift in continual semantic segmentation. 2021. 3

[13] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *Proc. Europ. Conf. Computer Vision*, 2020. 3

[14] Robert French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 1999. 3

[15] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. *Proc. Int. Conf. Machine Learning*, 2015. 2

[16] Behnam Gholami, Pritish Sahu, Ognjen Rudovic, Konstantinos Bousmalis, and Vladimir Pavlovic. Unsupervised multi-target domain adaptation: An information theoretic approach. *IEE Trans. Image Processing*, 2020. 3, 4

[17] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *J. Mach. Learn. Res.*, 2012. 2

[18] Tyler L. Hayes, Kushal Kafle, Robik Shrestha, Manoj Acharya, and Christopher Kanan. Remind your neural network to prevent catastrophic forgetting. In *Proc. Europ. Conf. Computer Vision*, 2020. 3

[19] Jianzhong He, Xu Jia, Shuaijun Chen, and Jianzhuang Liu. Multi-source domain adaptation with collaborative learning for semantic segmentation. In *Proc. Conf. Comp. Vision Pattern Rec.*, 2021. 2

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. Conf. Comp. Vision Pattern Rec.*, 2016. 7

[21] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A. Efros, and Trevor Darrell. Cycada: Cycle consistent adversarial domain adaptation. In *Proc. Int. Conf. Machine Learning*, 2018. 2

[22] Judy Hoffman, Dequan Wang, Fisher Yu, and Trevor Darrell. FCNs in the wild: Pixel-level adversarial and constraint-based adaptation. *arXiv:1612.02649*, 2016. 2

[23] Takashi Isobe, Xu Jia, Shuaijun Chen, Jianzhong He, Yongjie Shi, Jianzhuang Liu, Huchuan Lu, and Shengjin Wang. Multi-target domain adaptation with collaborative consistency learning. In *Proc. Conf. Comp. Vision Pattern Rec.*, 2021. 3, 4

[24] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proc. National Academy of Sciences*, 2017. 3

[25] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C Kot. Domain generalization with adversarial feature learning. In *Proc. Conf. Comp. Vision Pattern Rec.*, 2018. 2

[26] Z. Li and D. Hoiem. Learning without forgetting. *Proc. Europ. Conf. Computer Vision*, 2016. 3

[27] Ziwei Liu, Zhongqi Miao, Xingang Pan, Xiaohang Zhan, Dahua Lin, Stella X. Yu, and Boqing Gong. Open compound domain adaptation. In *Proc. Conf. Comp. Vision Pattern Rec.*, 2020. 2

[28] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *Proc. Int. Conf. Machine Learning*, 2015. 2

[29] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I. Jordan. Deep transfer learning with joint adaptation networks. In *Proc. Int. Conf. Machine Learning*, 2017. 2

[30] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Int. Conf. Neural Inf. Proc. Systems*, 2017. 3

[31] Toshihiko Matsuura and Tatsuya Harada. Domain generalization using a mixture of multiple latent domains. In *Proc. AAAI Conf. Artificial Intelligence*, 2020. 2

[32] Umberto Michieli and Pietro Zanuttigh. Incremental learning techniques for semantic segmentation. In *Proc. Int. Conf. Computer Vision Workshop*, 2019. 3

[33] Zak Murez, Soheil Kolouri, David J. Kriegman, Ravi Ramamoorthi, and Kyungnam Kim. Image to image translation for domain adaptation. *Proc. Conf. Comp. Vision Pattern Rec.*, 2018. 2

[34] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulò, and Peter Kortschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *Proc. Int. Conf. Computer Vision*, 2017. 6

[35] Van-Anh Nguyen, Tuan Nguyen, Trung Le, Quan Hung Tran, and Dinh Phung. Stem: An approach to multi-source domain adaptation with guarantees. In *Proc. Int. Conf. Computer Vision*, 2021. 2

[36] Prashant Pandey, Mrigank Raman, Sumanth Varambally, and Prathosh AP. Generalization on unseen domains via inference-time label-preserving target projections. In *Proc. Conf. Comp. Vision Pattern Rec.*, 2021. 2

[37] Kwanyong Park, Sanghyun Woo, Inkyu Shin, and In-So Kweon. Discover, hallucinate, and adapt: Open compound domain adaptation for semantic segmentation. In *Int. Conf. Neural Inf. Proc. Systems*, 2020. 2

[38] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *Workshop at Int. Conf. Neural Inf. Proc. Systems*, 2017. 6

[39] Xingchao Peng, Zijun Huang, Ximeng Sun, and Kate Saenko. Domain agnostic learning with disentangled representations. In *Proc. Int. Conf. Machine Learning*, 2019. 3

[40] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. In *Proc. Conf. Comp. Vision Pattern Rec.*, 2017. 3

[41] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *Proc. Europ. Conf. Computer Vision*, 2016. 6

[42] Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 1995. 3

[43] Mohammad Rostami. Lifelong domain adaptation via consolidated internal distribution. In *Int. Conf. Neural Inf. Proc. Systems*, 2021. 3

[44] Antoine Saporta, Tuan-Hung Vu, Matthieu Cord, and Patrick Pérez. Multi-target adversarial frameworks for domain adaptation in semantic segmentation. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 3, 4, 6, 7, 8

[45] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *Proc. Europ. Conf. Computer Vision*, 2016. 2

[46] Sebastian Thrun. Lifelong learning algorithms. In *Springer Learning to Learn*, 1998. 3

[47] Yi-Hsuan Tsai, Wei-Chih Hung, Samuel Schulter, Ki-hyuk Sohn, Ming-Hsuan Yang, and Manmohan Chandraker. Learning to adapt structured output space for semantic segmentation. In *Proc. Conf. Comp. Vision Pattern Rec.*, 2018. 2, 4

[48] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. *Proc. Conf. Comp. Vision Pattern Rec.*, 2017. 2

[49] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *CoRR*, 2014. 2

[50] Girish Varma, Anbumani Subramanian, Anoop Namboodiri, Manmohan Chandraker, and C V Jawahar. Dd: A dataset for exploring problems of autonomous navigation in unconstrained environments. In *Winter Conf. Appl. of Comp. Vision*, 2019. 6

[51] Riccardo Volpi, Diane Larlus, and Grégory Rogez. Continual adaptation of visual representations via domain randomization and meta-learning. In *Proc. Conf. Comp. Vision Pattern Rec.*, 2021. 3

[52] Tuan-Hung Vu, Himalaya Jain, Maxime Bucher, Matthieu Cord, and Patrick Pérez. ADVENT: Adversarial entropy minimization for domain adaptation in semantic segmentation. In *Proc. Conf. Comp. Vision Pattern Rec.*, 2019. 2, 4, 6, 7, 8

[53] Zuxuan Wu, Xintong Han, Yen-Liang Lin, Mustafa Gokhan Uzunbas, Tom Goldstein, Ser Nam Lim, and Larry S Davis. Dcan: Dual channel-wise alignment networks for unsupervised scene adaptation. In *Proc. Europ. Conf. Computer Vision*, 2018. 2

[54] Yanchao Yang and Stefano Soatto. Fda: Fourier domain adaptation for semantic segmentation. In *Proc. Conf. Comp. Vision Pattern Rec.*, 2020. 2

[55] Huanhuan Yu, Menglei Hu, and Songcan Chen. Multi-target unsupervised domain adaptation without exactly shared categories. *arXiv preprint arXiv:1809.00852*, 2018. 3, 4

[56] Shanshan Zhao, Mingming Gong, Tongliang Liu, Huan Fu, and Dacheng Tao. Domain generalization via entropy regularization. *Int. Conf. Neural Inf. Proc. Systems*, 2020. 2

[57] Sicheng Zhao, Bo Li, Xiangyu Yue, Yang Gu, Pengfei Xu, Runbo Hu, Hua Chai, and Kurt Keutzer. Multi-source domain adaptation for semantic segmentation. In *Int. Conf. Neural Inf. Proc. Systems*, 2019. 2