Plugging Self-Supervised Monocular Depth Into Unsupervised Domain Adaptation for Semantic Segmentation

This is the final peer-reviewed author's accepted manuscript (postprint) of the following publication:

Published Version:

Availability:
This version is available at: https://hdl.handle.net/11585/864973 since: 2022-02-23
Published:
DOI: http://doi.org/10.1109/WACV51458.2022.00206

Terms of use:
Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (https://cris.unibo.it/).
When citing, please refer to the published version.

(Article begins on next page)
Plugging Self-Supervised Monocular Depth into Unsupervised Domain Adaptation for Semantic Segmentation

Adriano Cardace  Luca De Luigi  Pierluigi Zama Ramirez  Samuele Salti  Luigi Di Stefano
Department of Computer Science and Engineering (DISI)
University of Bologna, Italy
{adriano.cardace2, luca.deluigi4, pierluigi.zama}@unibo.it

Abstract

Although recent semantic segmentation methods have made remarkable progress, they still rely on large amounts of annotated training data, which are often infeasible to collect in the autonomous driving scenario. Previous works usually tackle this issue with Unsupervised Domain Adaptation (UDA), which entails training a network on synthetic images and applying the model to real ones while minimizing the discrepancy between the two domains. Yet, these techniques do not consider additional information that may be obtained from other tasks. Differently, we propose to exploit self-supervised monocular depth estimation to improve UDA for semantic segmentation. On one hand, we deploy depth to realize a plug-in component which can inject complementary geometric cues into any existing UDA method. We further rely on depth to generate a large and varied set of samples to Self-Train the final model. Our whole proposal allows for achieving state-of-the-art performance (58.8 mIoU) in the GTA5 → CS benchmark. Code is available at https://github.com/CVLAB-Unibo/d4-dbst.

1. Introduction

Semantic segmentation is the task of classifying each pixel of an image. Nowadays, Convolutional Neural Networks can achieve impressive results in this task but require huge quantities of labelled images at training time [44, 3, 34, 41]. A popular trend to address this issue concerns leveraging on computer graphics simulations [42] or game engines [40] to obtain automatically synthetic images endowed with per-pixel semantic labels. Yet, a network trained on synthetic data only will perform poorly in real environments due to the so called domain-shift problem. In the last few years, many Unsupervised Domain Adaptation (UDA) techniques aimed at alleviating the domain-shift problem have been proposed in literature. These approaches try to minimize the gap between the labeled source domain (e.g. synthetic images) and the unlabeled target domain (e.g. real images) by either hallucinating input images, manipulating the learned features space or imposing statistical constraints on the predictions [58, 8, 65, 18].

At a more abstract level, UDA may be thought of as the process of transferring more effectively to the target domain the knowledge from a task solved in the source domain. This suggests that it may be possible to improve UDA by transferring also knowledge learned from another task to improve performance in the real domain. In fact, the existence of tightly related representations within CNNs trained for different tasks has been highlighted since the early works in the field [60], and it is nowadays standard practice to initialize CNNs deployed for a variety of diverse tasks, such as, e.g., object detection [46], semantic segmentation [4] and monocular depth estimation [14], with weights learned on Imagenet Classification [11]. The notion of transferability of representations among CNNs trained to solve different visual tasks has been formalized computationally by the Taskonomy proposed in [63]. Later, [38] has shown that it is possible to train a CNN to hallucinate deep features learned to address one task into features amenable to another task related to the former.

Inspired by these findings, we argue that monocular depth estimation could be an excellent task in order to gather additional knowledge useful to address semantic seg-
mentation in UDA settings. First of all, a monocular depth estimation network makes predictions based on 3D cues dealing with the appearance, shape, relative sizes and spatial relationships of the stuff and things observed in the training images. This suggests that the network has to predict geometry by implicitly learning to understand the semantics of the scene. Indeed, [37, 21, 24, 15] show that a monocular depth estimation network obtains better performances if forced to learn jointly a semantic segmentation task. We argue, though, the correlation between geometry and semantics to hold bidirectionally, such that a semantic segmentation network may obtain useful hints from depth information. This intuition is supported by [38], which shows that it is possible to learn a mapping in both directions between features learned to predict depth and per-pixel semantic labels. It is also worth observing how depth prediction networks tend to extract accurate information for regions characterized by repeatable and simple geometries, such as roads and buildings, which feature strong spatial and geometric priors (e.g., the road is typically a plane in the bottom part of the image) [13, 14, 47, 57]. Therefore, on one hand predicting accurately the semantics of such regions from depth information alone should be possible. On the other, a semantic network capable of reasoning on the geometry of the scene should be less prone to mistakes caused by appearance variations between synthetic and real images, the key issue in UDA for semantic segmentation.

Despite the above observations, injection of geometric cues into UDA frameworks for semantic segmentation has been largely unexplored in literature, with the exception of a few proposals, which either assume availability of depth labels in the real domain [56], a very restrictive assumption, or can leverage on depth information only in the synthetic domain due to availability of cheap labels [53, 27, 6]. In this respect, we set forth an additional consideration: nowadays effective self-supervised procedures allow for training a monocular depth estimation network without the need of ground-truth labels [14, 12, 70].

Based on the above intuitions and considerations, in this paper we propose the first approach that, thanks to self-supervision, allows for deploying depth information from both synthetic and unlabelled real images in order to inject geometric cues in UDA for semantic segmentation. Purposely, we adapt the knowledge learned to pursue depth estimation into a representation amenable to semantic segmentation by the feature transfer architecture proposed in [38]. As the geometric cues learned from monocular images yield semantic predictions that are often complementary to those attainable by current UDA methods, we realize our proposal as a depth-based add-on, dubbed D4 (Depth For), which can be plugged seamlessly into any UDA method to boost its performances, as illustrated in Fig. 1.

A recent trend in UDA for semantic segmentation is Self-Training (ST), which consists in further fine-tuning the trained network by its own predictions [72, 73, 68, 29, 33, 30]. We propose a novel Depth-Based Self-Training (DBST) approach which deploys once more the availability of depth information for real images in order to build a large and varied dataset of plausible samples to be deployed in the Self-Training (ST) procedure \(^1\).

Our framework can improve many state-of-the-art methods by a large margin in two UDA for semantic segmentation benchmarks, where networks are trained either on GTA5 [40] or SYNTHIA VIDEO SEQUENCES [42] and tested on Cityscapes [10]. Moreover, we show that our DBST procedure enables to distill the whole framework into a single ResNet101 [16] and achieve state-of-the-art performance. Our contributions can be summarized as follows:

- We are the first to show how to exploit self-supervised monocular depth estimation on real images to pursue semantic segmentation in a domain adaptation settings.
- We propose a depth-based module (D4) which can be plugged into any UDA for semantic segmentation method to boost performance.
- We introduce a new protocol (DBST) that exploits depth predictions to synthesize augmented training samples for the final self-training step deployed oftentimes in UDA for semantic segmentation pipelines.
- We show that leveraging on both D4 and DBST allows for achieving 58.8 mIoU in the popular GTA5→CS UDA benchmark, i.e., to the best of our knowledge, the new state-of-the-art.

2. Related Work

**Domain Adaptation.** Domain Adaptation is a promising way of solving semantic segmentation without annotations. Pioneering works [17, 58, 2, 9, 31, 66, 62, 28, 22] rely on CycleGANs [71] to convert source data into the style of target data, reducing the low-level visual appearance discrepancy among domains. Other works exploit adversarial training to enforce domain alignment [49, 50, 54, 67, 59, 36, 1, 52]. [55] extended this idea by aligning differently objects with low and high variability in terms of appearance. Few works tried to exploit depth information to boost UDA for semantic segmentation. [53], for example, proposes a unified depth-aware UDA framework that leverages the knowledge of depth maps in the source domain to perform feature space alignment. [43] extends this idea by modelling explicitly the relation between different visual semantic classes and depth ranges. [7], instead, considers depth as a way to obtain adaptation at both the input and output level. [56] is the first work to consider depth

\(^1\)See also [19] for concurrent work that proposes a similar idea.
in the target domain, although assuming supervision to be available. Conversely, we show how to deploy depth in the target domain without availability of ground-truth depths.

Self-Training. More recently, a new line of research focuses on self-training [26], where a semantic classifier is fine-tuned directly on the target domain, using its own predictions as pseudo-labels. [72, 73, 30] cleverly set class-confidence thresholds to mask wrong predictions. [69, 33, 68] propose to use pseudo-labels with different regularization techniques to minimize both the inter-domain and intra-domain gap. [64] instead, estimates the likelihood of pseudo-labels to perform online correction and denoising during training. Differently, [48] synthesizes new samples for the target domain by cropping objects from source images using ground truth labels and pasting them onto target images. Inspired by this work, we propose a novel algorithm for generating new samples to perform self-training on the target domain. In contrast to [48], our strategy is applied to target images only and relies on the availability of depth maps obtained through self-supervision.

Task Adaptation. All existing approaches tackle independently task adaptation or domain adaptation. [51] was the first paper to propose a cross-tasks and cross-domains adaptation approach, considering two image classification problems as different tasks. UM-Adapt [25] employs a cross-task distillation module to force inter-task coherency. Differently, [38], directly exploits the relationship among tasks to reduce the need for labelled data. This is done by learning a mapping function in feature space between two networks trained independently for two separate tasks, a pretext and target one. We leverage on this intuition but, unlike [38], our approach does not require supervision to solve the pretext task in the target domain.

3. Method

In Unsupervised Domain Adaptation (UDA) for semantic segmentation one wishes to solve semantic segmentation in a target domain, $D_T$, though labels are available only in another domain, referred to as source domain $D_S$. In the following we describe the two ingredients of our proposal to better tackle this problem. In Sec. 3.1 we show how to transfer information from self-supervised monocular depth to semantic segmentation and merge this knowledge with any UDA method (D4-UDA, Depth For UDA). Then, in Sec. 3.2 we introduce a Depth-Based Self-Training strategy (DBST) to further improve semantic predictions while distilling the whole framework into a single CNN.

3.1. D4 (Depth For UDA)

Semantics from Depth. The main intuition behind our work is that semantic segmentation masks obtained exploiting depth information have peculiar properties that make them suitable to improve segmentation masks obtained with standard UDA methods. However, predicting semantics from depth is an arduous task. Indeed, we experiment several alternatives (see Sec. 4.4 Alternative strategies to exploit depth) and find out that the most effective way is a procedure similar to the one proposed in [38], which we adapt to the UDA scenario. The pipeline works as follows: train one CNN to solve a first task on $D_S$ and $D_T$, train another CNN to solve a second task on $D_S$ only (i.e. the only domain where ground truth labels for the second task are available) and, finally, train a transfer function to map deep features extracted by the first CNN into deep features amenable to the second one. As the second CNN has been trained only on $D_S$, also the transfer function can be trained only on $D_S$ but, interestingly, it can generalize to $D_T$. As a consequence, at inference time one can solve the second task in $D_T$ based on the features transferred from the first task. We refer to [38] for further details.

Hence, if we assume the first and second task to consist in depth estimation and semantic segmentation, respectively, the idea of transferring features might be deployed in a UDA scenario since it gives the possibility to solve the second task on $D_T$ without the need of ground truth labels. However, the learning framework delineated in [38] assumes availability of ground-truth labels for the first task (depth estimation in our setting) also in $D_T$ (real images). As pointed out in Sec. 1, this assumption does not comply with the standard UDA for semantic segmentation problem formulation, which requires availability of semantic labels for source images ($D_S$) alongside with unlabelled target images only ($D_T$). To address this issue we propose to rely on depth proxy-labels attainable from images belonging to both $D_S$ and $D_T$ without the need of any ground-truth information. In particular, we propose to deploy one of the recently proposed deep neural networks, such as [14], that can be trained to perform monocular depth estimation based on a self-supervised loss that requires availability of raw image sequences only, i.e. without ground-truth depth labels. Thus, in our method we introduce the following protocol.
First, we train a self-supervised monocular depth estimation network on both $D_S$ and $D_T$. Then, we use this network to generate depth proxy-labels for both domains. We point out that we use such network as an off-the-shelf algorithm without the aim of improving depth estimation. Finally, according to [38], we train a first CNN to predict depth from images on both domains by the previously computed depth proxy-labels, a second CNN to predict semantic labels on $D_S$ and a transfer network which allows for predicting semantic labels from depth features in $D_T$. In the following, we will refer to such predictions as “semantics from depth” because they concern semantic information extracted from features amenable to perform monocular depth estimation.

**Combine with UDA.** Fig. 2 compares semantic predictions obtained from depth by the protocol described in the previous sub-section and from a recent UDA method. The reader may observe a clear pattern: predictions from depth tend to be smoother and more accurate on objects with large and regular shapes, like road, sidewalk, wall and building. However, they turn out often imprecise in regions where depth predictions are less informative, like thin things partially overlapped with other objects or fine-grained structures in the background. As UDA methods tend to perform better on such classes (see Fig. 3), we turn out often imprecise in regions where depth predictions are less informative, like thin things partially overlapped with other objects or fine-grained structures in the background. The D4 approach is designed to combine the semantic knowledge extracted from depth with that provided by any chosen UDA method in order to achieve more accurate semantic predictions. Depth information helps on large objects with regular shapes, which usually account for the majority of pixels in an image. On the contrary, UDA methods perform well in predicting semantic labels for categories that typically concern much smaller fractions of the total number of pixel in an image, like e.g. the traffic signs in Fig. 2. This orthogonality suggests that a simple yet effective way to combine the semantic knowledge drawn from depth with that provided by UDA methods consists in a weighted sum of predictions, with weights computed according to the frequency of classes in $D_S$ (the domain where semantic labels are available). As weights given to UDA predictions ($w_{uda}$) should be larger for rarer classes, they can be computed as:

$$w_{uda} = [w_{uda}^1, \ldots, w_{uda}^C] \quad \text{where} \quad w_{uda}^i = \frac{1}{\ln(\delta + f^i)} \quad (1)$$

where $C$ denotes the number of classes and $f^i = \frac{n^i}{n_{tot}}$ denotes their frequencies at the pixel level, i.e. the ratio between the number $n^i$ of pixels labelled with class $i$ in $D_S$ and the total number $n_{tot}$ of labelled pixels in $D_S$. Eq. 1 is the standard formulation introduced in [34] to compute bounded weights inversely proportional to the frequency of classes. We set $\delta$ in Eq. 1 to 1.02, akin to [34].

Accordingly, weights applied to semantic predictions drawn from depth ($w_{dep}$) are given by:

$$w_{dep} = [w_{dep}^1, \ldots, w_{dep}^C] \quad \text{where} \quad w_{dep}^i = 1 - w_{uda}^i \quad (2)$$

Thus, at each pixel of a given image we propose to combine semantics from depth and predictions yielded by any chosen UDA method as follows:

$$\hat{y}_f = w_{dep} \cdot \phi_T(\tilde{y}_{dep}) + w_{uda} \cdot \phi_T(\tilde{y}_{uda}), \quad (3)$$

where $\hat{y}_f$ is the final prediction, $\tilde{y}_{dep}$ and $\tilde{y}_{uda}$ are the logits associated with semantics from depth and the selected UDA method, respectively, $\phi_T$ denotes the softmax function with a temperature term $T$ that we set to 6 in our experiments.

As illustrated in Fig. 3, the formulation presented in Eq. 3 and symbolized as $\oplus$ can be used seamlessly to plug semantics obtained from self-supervised monocular depth into any existing UDA method. We will refer to the combination of a given UDA method with our D4 with the expression D4-UDA. Experimental results (Sec. 4.3) show that all recent s.o.t.a. UDA methods do benefit significantly from the complementary geometric cues brought in by D4.

Figure 3. Overview of our proposal. RGB images are first processed by two different segmentation engines to produce complementary predictions that are then combined by a weighted sum which accounts for the relative strengths of the two engines (Eq. 3). During the next step (DBST), predictions from D4-UDA, are used to synthesize augmented samples by mixing portions of different images according to depth and semantics. The augmented samples are used to train a final model, so as to distill the whole pipeline into a single network.
3.2. DBST (Depth-Based Self-Training)

We describe here our proposal to further improve semantic predictions and distill the knowledge of the entire system into a single network easily deployable at inference time. First, we predict semantic labels for every image in $D_T$ by our whole framework (i.e. D4 alongside a selected UDA method, referred to as D4-UDA); then, we use these labels to train a new model on $D_T$. This procedure, also known as self-training [26], has become popular in recent UDA for semantic segmentation literature [72, 73, 68, 29, 33, 30] and consists in training a model by its own predictions, referred to as pseudo-labels, sometimes through multiple iterations. On the other hand, we only perform one iteration, and the novelty of our approach concerns the peculiar ability to leverage on the depth information available for the images in $D_T$ to generate plausible new samples.

Running D4-UDA on $D_T$ yields semantic pseudo-labels for every image in $D_T$. Yet, as described in Sec. 3.1 (Semantics from Depth), each image in $D_T$ is also endowed with a depth prediction, provided by a self-supervised monocular depth estimation network. We can take advantage of this data to formulate a novel depth-aware data augmentation strategy whereby portion of images and corresponding pseudo-labels are copied onto others so as to synthesize samples for the self-training procedure. The crucial difference between similar approaches presented in [32, 48] and ours consists in the deployment of depth information to steer the data augmentation procedure towards more plausible samples. Indeed, a first intuition behind our method deals with semantic predictions being less accurate for objects distant from the camera: as such predictions play the role of labels in self-training, we prefer to pick closer rather than distant objects in order to generate training samples. Moreover, we reckon certain kinds of objects, like persons, vehicles and traffic signs, to be more plausibly transferable across different images as they tend to be small and less bound to specific spatial locations. On the contrary, it is quite unlikely to merge seamlessly a piece of road or building from a given image into a different one.

Given $N$ randomly selected images $x^n$ from $D_T$, with $n \in \{1, \ldots, N\}$, paired with semantic pseudo-labels $s^n$ and depth predictions $d^n$, we augment $x^1$, by copying on it pixels from the set $X^{n\in T} = \{x^2, \ldots, x^N\}$. For each pixel of the augmented image we have $N$ possible candidates, one from $x^1$ itself and $N - 1$ from the images in $X^{n\in T}$. We filter such candidates according to two criteria: the predicted depth should be lower than a threshold $t$ and the semantic prediction should belong to a predefined set of classes, $C^*$. Hence, we define the set of depths of the filtered candidates at each spatial location $p$ as:

$$D_p = \{d^n_p \mid d^n_p < t \land s^n_p \in C^* \} \quad n \in \{1, \ldots, N\}. \quad (4)$$

In our experiments, for each image the depth threshold $t$ is set to the $80^{th}$ percentile of the depth distribution, so as to avoid selecting pixels from the farthest objects in the scene. $C^*$ contains all things classes (e.g. person, car, traffic light, etc.), which include foreground elements that can be copied onto other images without altering the plausibility of the scene, while excluding all the stuff classes, which include background elements that cannot be easily moved across scenes. This categorization is similar to the one proposed in [55] and we consider it easy to replicate in other datasets.

Then, we synthesize a new image $x^2$ and corresponding pseudo-labels $s^2$, by assigning at each spatial location $p$ the candidate with the lowest depth, so that objects from different images will overlap plausibly into the synthesized one:

$$x^p_2 = x^p_1 \quad s^p_2 = s^p_1 \quad (5)$$

$$k = \begin{cases} 1 & D_p = \emptyset \\ n \succeq \min_{D_p \neq \emptyset} D_p & D_p \neq \emptyset \end{cases} \quad (6)$$

In Fig. 4 we depict our depth-based procedure to synthesize new training samples, considering, for the sake of simplicity, the case where $N = 2$.

Hence, with the procedure detailed above, we synthesize an augmented version of $D_T$, used to distill the whole D4-UDA framework into a single model by a self-training process. This dataset is much larger and exhibits more variability than the original $D_T$. Due to its reliance on depth information, we dub our novel technique as DBST (Depth-Based Self-Training). The results reported in Sec. 4.3 prove its remarkable effectiveness, both when used as the final stage following D4 as well as when deployed as a standalone self-training procedure applied to any other UDA method.

4. Experiments

4.1. Implementation Details

Network Architectures. We use Monodepth2 [14] to generate depth proxy-labels for the procedure described in
trained by SGD with batch size 2 for 70 epochs. The final
respectively, while the semantic segmentation network is
trained by Adam with 16GB of memory. In every training and test phase
Both networks consist of a backbone and an ASPP module. Both
networks are deployed on one NVIDIA Tesla V100 GPU. During DBST, we use
a simple 6-layers CNN with kernel size 3 × 3 and Batch Norm.
A common practice [49, 22, 28] and test our framework in the synthetic-to-real case using GT A5 [39, 40] or SYNTHIA [42] as synthetic datasets. The former consists in a large collection of video sequences of
real images, we leverage the popular Cityscapes dataset [35] as backbone.

**Training Details.** Our pipeline is implemented using PyTorch [35] and trained on one NVIDIA Tesla V100 GPU with 16GB of memory. In every training and test phase we resize input images to 1024 × 512, with the exception of
DBST, when we first perform random scaling and then random crop with size 1024 × 512. During DBST we use also color jitter to avoid overfitting on the pseudo-labels. In our version of [38], the depth and the transfer network are optimized by Adam [23] with batch size 2 for 70 and 40 epochs, respectively, while the semantic segmentation network is trained by SGD with batch size 2 for 70 epochs. The final
model obtained by DBST is trained again with SGD, batch size 3 and for 30 epochs. We adopt the One Cycle learning rate policy [45] in every training, setting the maximum learning rate to 10−2 but in DBST, where we use 10−3.

| Table 1. Results on GT A5 → CS.

<table>
<thead>
<tr>
<th>Method</th>
<th>Road</th>
<th>Sidewalk</th>
<th>Building</th>
<th>Wall</th>
<th>Fence</th>
<th>Pole</th>
<th>T-Light</th>
<th>Traffic Sign</th>
<th>Vegetation</th>
<th>Sky</th>
<th>Person</th>
<th>Rider</th>
<th>Car</th>
<th>Truck</th>
<th>Bus</th>
<th>Train</th>
<th>Make</th>
<th>Motorcycle</th>
<th>Bicycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdaptSegNet [49]</td>
<td>65.5</td>
<td>36.0</td>
<td>79.9</td>
<td>23.4</td>
<td>23.3</td>
<td>23.9</td>
<td>35.2</td>
<td>14.8</td>
<td>83.4</td>
<td>33.3</td>
<td>75.6</td>
<td>58.5</td>
<td>27.6</td>
<td>73.8</td>
<td>32.5</td>
<td>35.4</td>
<td>3.9</td>
<td>30.1</td>
<td>28.1</td>
</tr>
<tr>
<td>D4-AdaptSegNet + DBST</td>
<td>66.7</td>
<td>36.0</td>
<td>79.9</td>
<td>23.4</td>
<td>23.3</td>
<td>23.9</td>
<td>35.2</td>
<td>14.8</td>
<td>83.4</td>
<td>33.3</td>
<td>75.6</td>
<td>58.5</td>
<td>27.6</td>
<td>73.8</td>
<td>32.5</td>
<td>35.4</td>
<td>3.9</td>
<td>30.1</td>
<td>28.1</td>
</tr>
<tr>
<td>MaxSquare [44]</td>
<td>91.1</td>
<td>53.1</td>
<td>85.4</td>
<td>42.8</td>
<td>27.3</td>
<td>35.8</td>
<td>43.9</td>
<td>18.5</td>
<td>85.9</td>
<td>39.0</td>
<td>89.0</td>
<td>63.0</td>
<td>31.6</td>
<td>86.6</td>
<td>39.8</td>
<td>36.7</td>
<td>0</td>
<td>42.4</td>
<td>35.0</td>
</tr>
<tr>
<td>D4-MaxSquare + DBST</td>
<td>92.9</td>
<td>51.2</td>
<td>84.7</td>
<td>43.5</td>
<td>22.2</td>
<td>35.2</td>
<td>42.5</td>
<td>20.0</td>
<td>86.2</td>
<td>42.0</td>
<td>90.0</td>
<td>63.7</td>
<td>33.0</td>
<td>86.9</td>
<td>45.9</td>
<td>50.9</td>
<td>0</td>
<td>42.2</td>
<td>41.4</td>
</tr>
<tr>
<td>BDL [38]</td>
<td>68.2</td>
<td>34.7</td>
<td>84.2</td>
<td>24.6</td>
<td>27.6</td>
<td>30.2</td>
<td>30.0</td>
<td>36.0</td>
<td>85.0</td>
<td>43.6</td>
<td>83.0</td>
<td>58.6</td>
<td>41.6</td>
<td>83.1</td>
<td>35.3</td>
<td>49.7</td>
<td>3.3</td>
<td>28.8</td>
<td>28.6</td>
</tr>
<tr>
<td>D4-BDL + DBST</td>
<td>88.1</td>
<td>52.6</td>
<td>86.4</td>
<td>44.1</td>
<td>31.2</td>
<td>36.5</td>
<td>42.4</td>
<td>36.1</td>
<td>86.3</td>
<td>41.0</td>
<td>89.8</td>
<td>63.3</td>
<td>37.4</td>
<td>83.6</td>
<td>42.8</td>
<td>57.8</td>
<td>0</td>
<td>40.3</td>
<td>37.9</td>
</tr>
<tr>
<td>MNRENet [69]</td>
<td>91.5</td>
<td>35.0</td>
<td>84.6</td>
<td>24.0</td>
<td>36.8</td>
<td>44.1</td>
<td>42.7</td>
<td>36.5</td>
<td>85.0</td>
<td>36.5</td>
<td>82.3</td>
<td>63.1</td>
<td>44.4</td>
<td>85.8</td>
<td>32.9</td>
<td>38.2</td>
<td>2.0</td>
<td>27.1</td>
<td>41.8</td>
</tr>
<tr>
<td>D4-MNRENet + DBST</td>
<td>93.2</td>
<td>51.6</td>
<td>86.1</td>
<td>45.9</td>
<td>24.5</td>
<td>37.9</td>
<td>47.4</td>
<td>40.4</td>
<td>85.3</td>
<td>37.5</td>
<td>89.6</td>
<td>64.7</td>
<td>39.8</td>
<td>85.8</td>
<td>41.1</td>
<td>53.2</td>
<td>8.9</td>
<td>17.1</td>
<td>33.4</td>
</tr>
</tbody>
</table>

| Table 2. Results on SYNSEQ → CS.

<table>
<thead>
<tr>
<th>Method</th>
<th>Sky</th>
<th>Building</th>
<th>Road</th>
<th>Sidewalk</th>
<th>Wall</th>
<th>Fence</th>
<th>Pole</th>
<th>T-Light</th>
<th>Vegetation</th>
<th>Person</th>
<th>Rider</th>
<th>Car</th>
<th>Truck</th>
<th>Bus</th>
<th>Train</th>
<th>Motorcycle</th>
<th>Bicycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdaptSegNet* [49]</td>
<td>75.6</td>
<td>78.0</td>
<td>89.7</td>
<td>28.5</td>
<td>3.4</td>
<td>76.0</td>
<td>28.5</td>
<td>85.1</td>
<td>27.2</td>
<td>55.3</td>
<td>46.6</td>
<td>0</td>
<td>49.5</td>
<td>86.9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D4-AdaptSegNet + DBST</td>
<td>87.7</td>
<td>80.1</td>
<td>94.0</td>
<td>61.8</td>
<td>66.0</td>
<td>81.1</td>
<td>32.2</td>
<td>85.4</td>
<td>31.9</td>
<td>59.0</td>
<td>52.3</td>
<td>0</td>
<td>55.9</td>
<td>90.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MaxSquare* [44]</td>
<td>72.4</td>
<td>79.2</td>
<td>89.2</td>
<td>36.0</td>
<td>4.6</td>
<td>75.7</td>
<td>31.5</td>
<td>84.9</td>
<td>30.7</td>
<td>55.8</td>
<td>45.8</td>
<td>8.6</td>
<td>51.2</td>
<td>87.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D4-MaxSquare + DBST</td>
<td>87.5</td>
<td>80.0</td>
<td>93.7</td>
<td>61.8</td>
<td>7.3</td>
<td>80.8</td>
<td>33.2</td>
<td>84.6</td>
<td>35.1</td>
<td>48.1</td>
<td>48.1</td>
<td>2.8</td>
<td>56.5</td>
<td>90.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MNRENet* [69]</td>
<td>84.6</td>
<td>79.7</td>
<td>93.9</td>
<td>56.3</td>
<td>0</td>
<td>80.3</td>
<td>35.4</td>
<td>85.9</td>
<td>27.2</td>
<td>57.9</td>
<td>56.3</td>
<td>0</td>
<td>55.9</td>
<td>90.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D4-MNRENet + DBST</td>
<td>88.3</td>
<td>79.9</td>
<td>93.9</td>
<td>63.0</td>
<td>6.3</td>
<td>81.3</td>
<td>35.5</td>
<td>84.3</td>
<td>31.3</td>
<td>59.5</td>
<td>47.9</td>
<td>0</td>
<td>55.9</td>
<td>90.2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Sec. 3.1. We adapt the general framework presented in [38] to our setting by deploying the popular Deeplab-v2 [3] for depth estimation and semantic segmentation networks. Both networks consist of a backbone and an ASPP module [,], which, substitute, respectively, the encoder and decoder used in [35]. The backbone is implemented as a dilated ResNet50 [61]. We also remove the downsampling and upsampling operations used in [38] when learning the transfer function between depth and semantics. More precisely, in our architecture the transfer function is realized as a simple 6-layers CNN with kernel size 3 × 3 and Batch Norm [20]. Following the recent trend in UDA for semantic segmentation [49, 5, 28, 69, 55, 54, 22], during DBST we train a single Deeplab-v2 [3] model, with a dilated ResNet101 pre-trained on ImageNet [11] as backbone.

**Training Details.** Our pipeline is implemented using PyTorch [35] and trained on one NVIDIA Tesla V100 GPU with 16GB of memory. In every training and test phase we resize input images to 1024 × 512, with the exception of DBST, when we first perform random scaling and then random crop with size 1024 × 512. During DBST we use also color jitter to avoid overfitting on the pseudo-labels. In our version of [38], the depth and the transfer network are optimized by Adam [23] with batch size 2 for 70 and 40 epochs, respectively, while the semantic segmentation network is trained by SGD with batch size 2 for 70 epochs. The final model obtained by DBST is trained again with SGD, batch size 3 and for 30 epochs. We adopt the One Cycle learning rate policy [45] in every training, setting the maximum learning rate to 10−2 but in DBST, where we use 10−3.

4.2. Datasets

We briefly describe the datasets adopted in our experiments, pointing to the Suppl. Mat. for additional details. We follow common practice [49, 22, 28] and test our framework in the synthetic-to-real case using GT A5 [39, 40] or SYNTHIA [42] as synthetic datasets. The former consists in synthetic images captured with the game Grand Theft Auto V, while the latter is composed of images generated by rendering a virtual city. Since our method requires video sequences to train Monodepth2 [14], we use the split SYNTHIA VIDEO SEQUENCES (SYNTHIA-SEQ) in the experiments involving the SYNTHIA dataset. As for real images, we leverage the popular Cityscapes dataset [10], which consists in a large collection of video sequences of driving scenes from 50 different cities in Germany.

4.3. Results

We report here experimental results obtained in two domain adaptation benchmarks, which show how the combination with our D4 method allows to boost performance of recent UDA for semantic segmentation approaches.
GTA5→CS. Tab. 1 reports results on the most popular UDA benchmark for semantic segmentation, i.e. GTA5→CS, where methods are trained on GTA5 and tested on Cityscapes. We selected the most relevant UDA approaches proposed in the last years [49, 5, 28, 69, 55, 54, 22, 64], using checkpoints provided by authors when available. We report per-class and overall results in terms of mean intersection over union (mIoU) and pixel accuracy (Acc), when each method is either used stand-alone or deployed within our proposal (i.e. D4 + DBST). The reader may notice how every UDA method does improve considerably if combined with our proposal, despite the variability of their stand-alone performances. Indeed, AdaptSegNet [49], which yields about 42 in terms of mIoU, reaches 50 when embedded into our framework. Likewise, ProDA, currently considered the s.o.t.a. UDA method, improves in mIoU from 57.5 to 58.8. Moreover, we can observe in Tab. 1 that our method produces a general improvement for all classes, although we experience a certain performance variability for some of them (such as train, motorbike and bicycle), probably due to noisy pseudo-labels used during DBST. Conversely, our method yields consistently a significant gain on classes characterized by large and regular shapes, namely road, sidewalk, building, wall and sky. This validates the effectiveness of a) the geometric cues derivable from depth to predict the semantics of these kind of objects and b) the methodology we propose to leverage on these additional cues in UDA settings. This behavior is also clearly observable from qualitative in Fig. 5. We point out that, to the best of our knowledge, the performance obtained by D4-ProDA + DBST, i.e. 58.8 mIoU (last row of Tab. 1) establishes the new state-of-the-art for GTA5→CS.

SYNSEQ→CS. Akin to common practice in literature we present results also on the popular SYNTHIA dataset. As our pipeline requires video sequences to train the self-supervised monocular depth estimation network, we select the SYNTHIA VIDEO SEQUENCES split for training and the Cityscapes dataset for testing. We will call this setting SYNSEQ→CS. To address it, we re-trained the UDA methods for which the code is available and the training procedure is more affordable in terms of memory and run-time requirements, namely AdaptSegNet [49], MaxSquare [5] and MRNET [69]. The results in Tab. 2 show that all the selected UDA approaches exhibit a substantial performance gain when coupled with our proposal, with a general improvement in all classes. In particular, similarly to the results obtained in GTA5→CS, we observe a consistent improvement for classes related to objects with large and regular shapes (as depicted also in Fig. 5), with the only exception of a slight performance drop for the class building when using MRNET [69] (last row of Tab. 2). We argue that our approach is relatively less effective with MRNET [69] as, unlike AdaptSegNet [49] and MaxSquare [5], it yields already satisfactory results in those classes which are usually improved by the geometric clues injected by D4.

In the Suppl. Mat. we show that it is also possible to exploit the depth ground-truths provided by the SYNTHIA dataset as an additional source of supervision during the training of Monodepth2 [14], obtaining a small improvement in the performances of the overall framework.

4.4. Analysis

We report here the most relevant analysis concerning our work. Additional ones can be found in the Suppl. Mat..

Ablation studies. In Tab. 3, we analyze the impact on the performance of our two main contributions, i.e. injec-
tion of geometric cues into UDA methods by D4 and DBST. 

Purposely, we select the GTAs5→CS benchmark and, for the top performing UDA methods, we report the mIoU figures obtained by using the stand-alone UDA method (column UDA), combining it with D4 (column D4-UDA), applying DBST directly on the stand-alone method (column UDA + DBST) and embedding the method into our full pipeline (column D4-UDA + DBST). We can observe that each of our novel contributions improves the performance of the most recent UDA methods by a large margin, which is even more remarkable considering that the selected methods already include one or more step of self-training. Moreover, D4 and DBST further enhance the performances of any selected method when deployed jointly, as shown in the column D4-UDA + DBST, suggesting that they are complementary. In order to further assess the effectiveness of DBST, in the column D4-UDA + ST we report results obtained by D4-UDA in combination with a baseline self-training procedure, which consists in simply fine-tuning the model by its own predictions on the images of the target domain. As the only difference between this procedure and our DBST is the dataset employed for fine-tuning, the results prove the effectiveness of DBST in generating a varied set of plausible samples more amenable to self-training than the original images belonging to the target domain.

Alternative strategies to exploit depth. As explained in Sec. 3.1 Semantics from Depth, we rely on the mechanism of transferring features across tasks and domains from [38] to inject depth cues into semantic segmentation. To validate our choice, we explore two possible alternatives, namely DeepLabV2-RGBD and DeepLabV2-Depth. Both consist in the popular DeepLabV2 [3] network, with RGBD images in input in the first case and depth maps (no RGB) in the second (more details in the Suppl. Mat.). Tab. 4 compares the performance of these alternatives with our method, either when used standalone (rows 2, 3, and 4) or when combined with LTIR [22] according to the strategy presented in Sec. 3.1 Combine with UDA. Results allow us to make some important considerations. First, our intuition on the possibility of exploiting depth to improve semantics is correct since also simple approaches improve over the baseline (reported in the first row of the table). Nonetheless, these naive methods produce a significantly smaller improvement compared to our approach, showing that our decision to adapt [38] to the UDA scenario is not obvious. Moreover, [38] requires only RGB images at test time. Finally, when combined with LTIR [22], a stronger depth-to-semantic model provides better results, validating our choice once again.

Impact of video sequences. As described in Sec. 3.1, we obtain depth proxy-labels with a self-supervised depth estimation network [14], that we train using the raw video sequences (just RGB images) provided by the datasets involved in our experiments. In order to validate that using video sequences from the target domain doesn’t provide any advantage to our framework, we train AdaptSegNet [49] on GTAs5→CS using the whole training split available for Cityscapes (i.e. 83300 images with temporal consistency). We choose AdaptSegNet [49] since it can be considered the building block of many UDA methods. We observe a drop in performances from 42.4 to 41.9 mIoU, showing that using video sequences does not boost semantic segmentation in a UDA setting, probably because of the similarity between consecutive frames, and that the improvement produced by our framework is provided by the effective strategy that we adopt to exploit depth.

5. Conclusion

We have shown how to exploit self-supervised monocular depth estimation in UDA problems to obtain accurate semantic predictions for objects with strong geometric priors (like road and buildings). As all recent UDA approaches lack such geometric knowledge, we build our D4 method as a depth-based add-on, pluggable into any UDA method to boost performances. Finally, we employed self-supervised depth estimation to realize an effective data augmentation strategy for self-training. Our work highlights the possibility of exploiting auxiliary tasks learned by self-supervision to better tackle UDA for semantic segmentation, paving the way for novel research directions.
References


[38] Pierluigi Zama Ramirez, Alessio Tonioni, Samuele Salti, and Luigi Di Stefano. Learning across tasks and domains. 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Oct 2019. 1, 2, 3, 4, 6, 8


[42] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016. 1, 2, 6


[49] Yi-Hsuan Tsai, Wei-Chih Hung, Samuel Schulte, Kihyuk Sohn, Ming-Hsuan Yang, and Mannohman Chandraker. Learning to adapt structured output space for semantic segmentation. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Jun 2018. 2, 6, 7, 8


[58] Zuxuan Wu, Xintong Han, Yen-Liang Lin, Mustafa Gökhan Uzunbas, Tom Goldstein, Ser Nam Lim, and Larry S. Davis. Dean: Dual channel-wise alignment networks for unsupervised scene adaptation. Lecture Notes in Computer Science, page 535–552, 2018. 1, 2


1. Additional Implementation Details

As stated in Sec. 3.1 of the main paper, we obtain depth proxy-labels by deploying a self-supervised method for solving monocular depth estimation from video sequences. Specifically, we train Monodepth2 [6] following the training protocol and hyper-parameters used in the original paper. We train it for 20 epochs using mixed mini-batch of size 6, composed of 3 real and 3 synthetic images. We resize samples at resolution $1024 \times 512$ for training and testing. It is important to train the network on both domains jointly because we want depth predictions to be aligned across domains. Self-supervised depth methods typically estimates depth maps up to a scale-factor. Thus, we train on both domains simultaneously to force the network to yield predictions from the two domains that share the same range and scale. When $D_S$ is synthetic, we can collect depth ground-truth labels with minimum effort. In such case, we could exploit these labels to provide an additional source of supervision to Monodepth2. SYNTHIA-SEQ provides much less images with smaller variability with respect to GTA5, but provides depth ground-truth labels. In the SYNEQ→CS setting, we could train Monodepth2 by adding a $L_1$ loss between predictions and ground-truths of SYNTHIA-SEQ to the set of Monodepth2 losses, so as to achieve better pseudo-labels results. Nevertheless, the availability of ground-truth labels is not crucial to improve the performance of the considered UDA method. Indeed, in Tab. 1 we can observe that the use of synthetic depth ground-truth labels provides just a slight performance improvement (i.e. 1% mIoU or less).

As regards the training of semantics prediction from depth features, we follow the protocol explained in [9]. We train the depth network simultaneously on $D_S$ and $D_T$, by minimizing the mean absolute error (i.e. $L_1$ loss) between predicted depth maps and depth proxy-labels, previously generated for both domains. Then, we train the semantic network only on $D_S$, using a weighted Cross Entropy loss with weights computed as in [18]. The weights of the two networks are pre-initialized on ImageNet, and, following a common protocol [14, 18, 7], all Batch Normalization layers are frozen both at training and test time to use ImageNet statistics. Differently from [9], we deploy the more performant DeepLabV2 [1] architecture for both networks: the framework requires to split the network into an encoder and a decoder, we consider the backbone as the encoder and the ASPP module as the decoder. Hence, the transfer function in D4 is learned by minimizing the mean squared distance (i.e. $L_2$ loss) between the semantic features extracted by the semantic network encoder and the ones hallucinated by the transfer function itself starting from the depth encoder. Finally, during DBST, the final distilled model is obtained by minimizing a standard Cross Entropy loss on $D_T$ and exploiting only the pseudo-labels, as explained in Sec. 3.2 of the main paper.

2. Additional Datasets Details

Cityscapes. The Cityscapes dataset [4] provides a large collection of video sequences of driving scenes from 50 different European cities. The dataset is composed of 150000 video-sequence images, of which 83300 are used for training. A subset of 5000 images from Cityscapes is commonly used as benchmark for semantic segmentation, as these images are annotated with high-quality pixel-level semantic labels (19 classes). This subset is split into train, validation and test with 2975, 500 and 1525 images respectively. In our experiments we train Monodepth2 [6] on the 83300 training sequences. For training D4 and DBST we use the 2975 train images (without their semantic labels) and, following the protocol adopted in recent works [14, 2, 8, 18, 17, 16, 7], we evaluate our final model on the validation split. The augmented dataset obtained during DBST starting from the 2975 images accounts for 7500 samples.

GTA5. The GTA5 dataset [10, 11] consists in synthetic
images captured while playing the video-game Grand Theft Auto V. It consists of 120000 video-sequence images that we use in the Monodepth2 [6] training procedure. Moreover, the dataset provides 24966 samples with fine semantic annotations (same 19 classes as Cityscapes). We train the depth network of D4 on only 3000 randomly sampled images among the 24966 to keep the training balanced with the 2975 images of Cityscapes. Finally, we train the semantic and transfer network of D4 on the whole 24966 synthetic images.

SYNTHIA VIDEO SEQUENCES. The SYNTHIA dataset [12] is composed of images generated by rendering a virtual city created with the Unity development platform. Since our method requires video sequences to train Monodepth2 [6], we use the split SYNTHIA VIDEO SEQUENCES, selecting sub-sequences Spring, Summer, Fall, Winter, Dawn and Fog. We collect thus a total of 26948 images, paired with fine-grained semantic labels (12 classes in common with Cityscapes). In particular, we train on sky, building, road, sidewalk, fence, vegetation, pole, car, traffic sign, person, bicycle, traffic light. It is worth noticing that to make the Cityscapes dataset consistent with SYNTHIA VIDEO SEQUENCES, it is necessary to map the Cityscapes class rider into bicycle and collapse bus and truck into car. We use only 3000 randomly sampled images to train the depth, semantic and transfer network of D4, as well as for the training of the other considered methods which were retrained by us (* in Tab. 2 of the main paper) due to the authors not providing their results on SYNTHIA VIDEO SEQUENCES.

3. Semantics From Depth

In this section, we evaluate alternative ways to predict semantics in the target domain by exploiting also the depth cues available once depth proxy-labels have been computed as discussed in sec 3.1 (Semantics from depth) of the main paper. This study motivates our choice to rely on the mechanism of transferring features across tasks and domains [9], with the improvements and modifications discussed in Sec. 4.1 of the main paper and Sec. 1 of this supplementary document. As we have semantic labels only for the source domain $D_S$, all approaches are trained only on $D_S$, and their ability to generalize is assessed on the target domain $D_T$.

We investigate two possible alternatives, namely:

- a semantic segmentation network that processes RGB-D images, where the proxy depth of each image is stacked as an additional channel
- a semantic segmentation network that processes directly proxy depths, without using RGB information.

We realize both options by training the popular DeepLabV2 [1] architecture to perform semantic segmenta-

<table>
<thead>
<tr>
<th>Method</th>
<th>mIoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdaptSegNet* [14]</td>
<td>49.5</td>
</tr>
<tr>
<td>D4-AdaptSegNet + DBST (w/o synthetic GT)</td>
<td>55.9</td>
</tr>
<tr>
<td>D4-AdaptSegNet + DBST (w/ synthetic GT)</td>
<td>56.9</td>
</tr>
<tr>
<td>MaxSquare* [2]</td>
<td>51.2</td>
</tr>
<tr>
<td>D4-MaxSquare + DBST (w/o synthetic GT)</td>
<td>56.5</td>
</tr>
<tr>
<td>D4-MaxSquare + DBST (w/ synthetic GT)</td>
<td>57.4</td>
</tr>
<tr>
<td>MRNET* [18]</td>
<td>54.5</td>
</tr>
<tr>
<td>D4-MRNET + DBST (w/o synthetic GT)</td>
<td>55.9</td>
</tr>
<tr>
<td>D4-MRNET + DBST (w/ synthetic GT)</td>
<td>56.3</td>
</tr>
</tbody>
</table>

Table 1. Results on the SYNSEQ→CS benchmark with or without synthetic ground-truths. * denotes method retrained by us.

In Tab. 2 we report mIoU results obtained on Cityscapes (i.e. our target domain) by DeepLabV2-RGB, DeepLabV2-RGBD, DeepLabV2-Depth, and our method. We observe that the RGBD and the Depth versions yield slightly better results compared to the RGB baseline. Interestingly, DeepLabV2-Depth provides better results than DeepLabV2-RGB and DeepLabV2-RGBD, which supports our intuition about semantic cues extracted from depth alone being more effectively transferable across different domains due to their reliance on geometry rather than appearance. Yet, the ability to overcome the domain shift by DeepLabV2-RGBD and DeepLabV2-Depth is limited, as performance is low for both variants. On the contrary, by tackling the problem with the method proposed in the main paper, we can improve the baseline by 8.6% in terms of mIoU.

Moreover, we evaluate DeepLabV2-RGBD and DeepLabV2-Depth also in combination with an UDA method, as proposed in Sec. 3.1 (Combine with UDA) of the main paper. In the last three rows of Tab. 2, we report mIoU results obtained by such combinations (row 5 and 6), compared to our proposal (last row), while considering one of the best performing UDA methods, namely LTIR [7]. As intuitively expected, we observe that a better depth-based semantic model leads to a better combination with the selected UDA method, motivating once again the need for an approach robust to domain-shift in order to infer semantics from depth cues in UDA settings.

Rather than relying on self-supervised depth on both do-
mIoU

<table>
<thead>
<tr>
<th>Method</th>
<th>mIoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeepLabV2 RGB</td>
<td>34.5</td>
</tr>
<tr>
<td>DeepLabV2-RGBD</td>
<td>35.5</td>
</tr>
<tr>
<td>DeepLabV2-Depth</td>
<td>36.5</td>
</tr>
<tr>
<td>Semantics from depth (sec 3.1)</td>
<td>43.1</td>
</tr>
<tr>
<td>DeepLabV2-RGBD ⊕ LTIR [7]</td>
<td>47.7</td>
</tr>
<tr>
<td>DeepLabV2-Depth ⊕ LTIR [7]</td>
<td>49.3</td>
</tr>
<tr>
<td>D4-LTIR (i.e. Semantics from depth ⊕ LTIR)</td>
<td>51.1</td>
</tr>
</tbody>
</table>

Table 2. Comparison between alternative methods to infer semantics with the aid of depth cues. DeepLabV2-RGB, DeepLabV2-RGBD and DeepLabV2-Depth stand for DeepLabV2 [1] trained on $D_S$, using respectively RGB images, RGBD images or depth proxy-labels as input, while “Semantics from depth” is the approach described in the subsection with the same name of sec 3.1 in the main paper. The symbol $\oplus$ represents the merge operation described in subsection Combine with UDA of Sec. 3.1 of the main paper. Results are reported in terms of mIoU on the Cityscapes dataset.

mains as done for the previous cases, one may try to use just the depth provided by synthetic source dataset. To the best of our knowledge, only two works [15, 3] proposed to exploit depth in a UDA context for outdoor scenes segmentation. We compare here our D4 module with [15], the only publicly available framework, to show that the additional information for the target domain is a key component for Domain Adaptation. We retrained [15] with the same hyper-parameters, and changed only the training split (i.e. SYNTIIHIA-SEQ instead of SYTNIIHA-RAND-CITYSCAPES). As Tab 3 shows, D4 surpasses by a large margin (3.6%) [15], suggesting that self-supervised information for the target domain can be used to boost performance in Domain Adaptation.

<table>
<thead>
<tr>
<th>Method</th>
<th>mIoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>DADA [15]</td>
<td>42.3</td>
</tr>
<tr>
<td>D4 (ours)</td>
<td>45.9</td>
</tr>
</tbody>
</table>

Table 3. Comparison between depth-based frameworks.

4. DBST vs DACS [13]

In Tab. 4 we compare our DBST with the method presented in DACS [13], as they share some similarities. In particular, both approaches generate training samples by copying portions of images onto other images. However, they differ in three main aspects:

- [13] copies portions of images from $D_S$ onto images from $D_T$, while in our DBST we use exclusively images from $D_T$.
- In our proposal, we copy only image patches whose semantic predictions belong to a predefined set of classes that we deem as more amenable to be moved across images, like, e.g., person, car and pole; conversely, in [13] no semantic filter is applied to select the patches that will be copied across the images.
- Unlike [13], we exploit depth information to plausibly stack objects in the generated sample.

In addition to these points, in our DBST we further exploit depth information to guide the selection of the patches to be copied by excluding areas of the scene that are too far away from the camera, where semantic predictions are less likely accurate. In Tab. 4 we report results in the GTA5→CS benchmark when applying DBST or [13] to D4 combined with [7]: our DBST outperforms the strategy proposed in [13], though the latter can also yield a notable performance improvement.

5. Adding videos to UDA methods

In this section, we empirically demonstrate that using additional raw information is not directly useful for the UDA setting in semantic segmentation. To this purpose, we adopt [14], which makes use of adversarial training and it can be considered as the main building block of many UDA methods proposed in the literature. Moreover, adversarial training is a plausible strategy to exploit additional unlabeled images for the target domain. Driven by this reasoning, we retrained [14] in the GTA5→CS benchmark using the whole training split available in Cityscapes (i.e. 83300 images with temporal consistency). The result reported in Tab. 5 suggests that simply collecting more data is not enough to boost semantic semantic segmentation in a UDA setting, and more advanced techniques as the one proposed in this work are necessary to extrapolate useful data.

<table>
<thead>
<tr>
<th>Method</th>
<th>mIoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>D4-LTIR [7]</td>
<td>51.1</td>
</tr>
<tr>
<td>D4-LTIR [7] + DBST</td>
<td>54.1</td>
</tr>
</tbody>
</table>

Table 4. Comparison between the approach proposed in [13] (DACS) and our DBST, when applied to our D4 combined with [7]. Results are reported in terms of mIoU in the GTA5→CS benchmark.

<table>
<thead>
<tr>
<th>Method</th>
<th>mIoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdaptSegNet (w/o video) [14]</td>
<td>42.4</td>
</tr>
<tr>
<td>AdaptSegNet (w/ video)</td>
<td>41.9</td>
</tr>
</tbody>
</table>

Table 5. AdaptSegNet [14] trained with or without additional unlabeled target images.
6. Qualitative Results

In Fig. 1, 2, 3, 4, 5, 6 we report several qualitative results of our D4 proposal combined with the different UDA methods reported in Tab. 1 and Tab. 2 of the main paper. In every case, we observe an overall improvement in the quality of the predictions. In particular, thanks to the additional information provided by depth maps, the errors in large objects with regular shapes are partially removed (see first and second column of Fig. 1). Moreover, with the proposed merging algorithm (Sec 3.1) and with the DBST algorithm detailed in Sec. 3.2, we also preserve the good performance of the selected UDA method for certain classes. For instance, all the predictions concerning classes such as pole and traffic sign are always maintained or even improved (see second row of Fig. 2).

7. DBST - Qualitative Results

In Fig. 7 and 8 we show some training samples obtained with our DBST algorithm. As explained in Sec. 3.2 of the main paper, we use multiple images from $D_T$ as source, alongside with the corresponding depth maps and predictions (referred to as pseudo-labels), to synthesize new training pairs. We can notice how the newly generated samples contain a lot of patterns that would not be present in the original images, enabling a more effective Self-Training procedure. We also point out how, thanks to the use of depth maps, the generated pairs look realistic. For example, in the third row of Fig. 7, the rider on the left side of the image is pasted in front of the pole since it appears closer in the depth maps of the two images.

8. Depth Proxy-Labels

Fig. 9, 10, 11 report depth proxy-labels obtained in the first step of our pipeline by the self-supervised approach proposed in Monodepth2 [6]. We note how the produced depth maps are smooth and accurate on the static parts of the scene (such as road and buildings), while they tend to be noisy on moving objects (like cars and pedestrians). Despite these imperfections, depth proxy-labels produced by [6] provide a solid base of geometric clues for objects with large and regular shapes, which are extensively exploited in our proposal.

References


[14] 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. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Jun 2018. 1, 2, 3, 6, 8


Figure 1. Qualitative results in the GTA5→CS benchmark. From left to right: RGB, prediction from AdaptSegNet [14], prediction from D4-AdaptSegNet + DBST (our proposal), Ground-Truth.

Figure 2. Qualitative results in the GTA5→CS benchmark. From left to right: RGB, prediction from MaxSquare [2], prediction from D4-MaxSquare + DBST (our proposal), Ground-Truth.
Figure 3. Qualitative results in the GTA5→CS benchmark. From left to right: RGB, prediction from LTIR [7], prediction from D4-LTIR + DBST (our proposal), Ground-Truth.

Figure 4. Qualitative results in the GTA5→CS benchmark. From left to right: RGB, prediction from BDL [8], prediction from D4-BDL + DBST (our proposal), Ground-Truth.
Figure 5. Qualitative results in the SYNSEQ→CS benchmark. From left to right: RGB, prediction from AdaptSegNet [14], prediction from D4-AdaptSegNet + DBST (our proposal), Ground-Truth.

Figure 6. Qualitative results in the SYNSEQ→CS benchmark. From left to right: RGB, prediction from MaxSquare [2], prediction from D4-MaxSquare + DBST (our proposal), Ground-Truth.
Figure 7. RGB and pseudo-labels generated for our DBST procedure using D4-LTIR in the GTA5→CS benchmark.
Figure 8. RGB and pseudo-labels generated for our DBST procedure using D4-MRNET in the SYNSEQ→CS benchmark.
Figure 9. Depth proxy-labels for the Cityscapes dataset obtained with Monodepth2 [6]. From top to bottom: RGB, depth obtained by training Monodepth2 on Cityscapes and GTA5 sequences, depth obtained by training Monodepth2 on Cityscapes and SYNTHIA-SEQ sequences. Depth maps are shown as inverse depth maps for a better visualization.

Figure 10. Depth proxy-labels for the GTA5 dataset obtained with Monodepth2 [6]. We show RGB images (first row) and corresponding depth maps (second row), shown as inverse depth maps for a better visualization.

Figure 11. Depth proxy-labels for the SYNTHIA-SEQ dataset obtained with Monodepth2 [6]. We show RGB images (first row) and corresponding depth maps (second row), shown as inverse depth maps for a better visualization.