Maximizing Cosine Similarity Between Spatial Features for Unsupervised Domain Adaptation in Semantic Segmentation

Inseop Chung[1*] Daesik Kim[2] Nojun Kwak[1]
[1]Seoul National University [2]NAVER WEBTOON Corp.
{jis3613, nojunk}@snu.ac.kr daesik.kim@webtoonscorp.com

Abstract

We propose a novel method that tackles the problem of unsupervised domain adaptation for semantic segmentation by maximizing the cosine similarity between the source and the target domain at the feature level. A segmentation network mainly consists of two parts, a feature extractor and a classification head. We expect that if we can make the two domains have small domain gap at the feature level, they would also have small domain discrepancy at the classification head. Our method computes a cosine similarity matrix between the source feature map and the target feature map, then we maximize the elements exceeding a threshold to guide the target features to have high similarity with the most similar source feature. Moreover, we use a class-wise source feature dictionary which stores the latest features of the source domain to prevent the unmatching problem when computing the cosine similarity matrix and be able to compare a target feature with various source features from various images. Through extensive experiments, we verify that our method gains performance on two unsupervised domain adaptation tasks (GTA5→Cityscapes and SYNTHIA→Cityscapes).

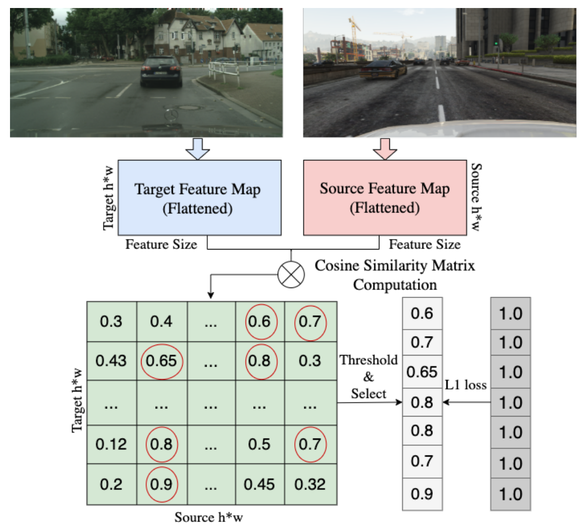

Figure 1. Overview of our cosine similarity loss at the feature level. By computing the cosine similarity matrix, we can compare which target feature is similar to which source feature and selectively maximize the cosine similarity between them spatial-wise. Here, the threshold is set to 0.5. Broad arrows on top indicate that the images are forwarded through the feature extractor of the segmentation network.

1. Introduction

Semantic segmentation [14] is a pixel-wise classification task which segments an image based on semantic understanding. Recently, its progress has been significantly driven by deep convolutional neural networks. However, training a segmentation network requires dense pixel-level annotations which are laborious, time-consuming and expensive. Unsupervised Domain Adaptation (UDA) for semantic segmentation is one possible way to solve this problem. It adapts a model trained on a dataset with labels (source domain) to another dataset without labels (target domain). The source and the target domain datasets share common classes and environments, thus it is possible to adapt between the domains. Typically, as source domain datasets, synthetically generated computer graphic images such as GTA5 [17] and SYNTHIA [18] datasets which are relatively easy to get the annotations are used. On the other hand, a target domain dataset consists of real images, for example, Citycapses [5] dataset for which the annotations are hard to obtain.

Most of current UDA methods employ adversarial adaptation to overcome the domain discrepancy. It minimizes the discrepancy by fooling a discriminator network that is trained to distinguish the originating domain of an image correctly. However, it has a critical drawback that it only sees the output globally and checks whether it is from the source or the target domain, making it into a binary classification problem. This is not suitable to adapt target features to specific source features that are most similar. Wang et al. [24] pointed out these problems and showed that using ad-

*This work is conducted when the author was in research internship at NAVER WEBTOON Corp.

versarial loss can severely impair the performance in long-term training. Instead, they proposed a method of aligning the source and the target feature distributions by minimizing the L1 loss between the average feature representations of the two domains while treating the 'stuff' and the 'instance' classes differently. It is claimed that they could overcome the instability of adversarial adaptation and shift the target features towards the most similar source features. Nonetheless, this method has a couple of downsides that it computes an average of features for each class which loses spatial information and it still uses the adversarial adaptation.

In this paper, we further investigate and develop in this direction of aligning the feature distributions by maximizing the cosine similarity between the two domains at the feature level. Moreover, we show that adversarial adaptation is unnecessary and can be replaced by our newly-proposed method. Our intuition is that if the features of the two domains have high cosine similarity, their predictions would also be very similar. The contributions of our work are as follows. First, instead of taking an average of features for each class as in [24], we compute a cosine similarity matrix to measure how a target feature is similar to each source feature spatial-wise. As can be seen in Fig. 1, a cosine similarity matrix is computed between the flattened feature maps of the source and the target domain with respect to the feature dimension, producing a 2D matrix whose dimension is target's spatial size (height $\times$ width) by source's spatial size. Each row represents how a target feature is similar to each source feature along the spatial dimension. In practice, a feature map is split into classes so the cosine similarity matrix is computed for each class. From the matrix, we selectively maximize elements that have higher cosine similarity than a pre-defined threshold so that they become closer to 1. We believe that if a target feature is similar to a source feature, with their cosine similarity being higher than a certain threshold, those two features belong to the same semantic information (the same class). We call this as 'cosine similarity loss' and this ensures the target domain features to be aligned with the source domain feature of the same class. Our method nudges a target feature to the most similar source features of the same class. It suggests that even if the target and the source features belong to the same class, the target features have to be selectively closer to the source features that actually have high semantic similarity.

Second, we use a dictionary that maintains the latest source features. Our method splits a source feature map by classes and stores them to a dictionary. The keys of the dictionary are the class identities and its values are the source features belonging to each class. The source features are stored as a queue, thus only the newest source features are kept for each class. The target feature map is also split by classes using either the pseudo-label or the prediction output. Then we compute the cosine similarity matrix between the split target features and the source features stored in the dictionary for each class. This approach enables to compare the target features with more variety of source features from various images. It also solves the unmatching problem which occurs when a certain class only appears in the current target image and not in the current source image thus the target features of the class do not have any source features to be maximized with.

For the last, we do not utilize the adversarial adaptation loss which is known to be complex and difficult to train and not appropriate to adapt target features to the most similar source features. We found that it is unnecessary and rather contradicts with our cosine similarity loss because it disrupts the training when used together. We empirically show that it can be replaced by our method. Therefore, we train only with the segmentation loss and our cosine similarity loss. We evaluate our method on two UDA of semantic segmentation benchmarks, 'GTA5 $\rightarrow$ Cityscapes' and 'SYNTHIA $\rightarrow$ Cityscapses' and show that our method has a valid performance gain.

2. Related Work

The goal of unsupervised domain adaptation (UDA) is to adapt and utilize the knowledge a model has learned from the source domain to perform well on the target domain without the supervision of target domain labels. This problem is challenging due to the discrepancy caused by the domain shift. UDA in classification is widely studied and has shown great progress [29], on the other hand, UDA of semantic segmentation is more challenging sine it is a pixel-wise classification task. Many works have been proposed in a variety of directions but we categorize them into three methodologies, image translation, adversarial adaptation and self-supervised learning via pseudo-labels.

Image translation method closes the domain gap at the image level by style-transferring source domain images into target domain to maximize the visual similarity [8, 14, 13]. It tries to apply the style factors of the target domain to the source domain. Some works employ an image translation algorithm such as CycleGAN [30]. [25] uses an image generator network to produce new synthesized source image via channel-wise feature alignment. [15] proposes an adaptive image translation method which uses semantic output of the network. [28] shows a way to translate the source images without such complex translation network using Fourier transform and its inverse. [12] diversifies the texture of the source domain by style-transfer and train the segmentation network to learn texture-invariant representation.

Adversarial adaptation tries to match the distribution of the source and the target domains at the feature and the prediction output level. It uses the adversarial training proposed in GAN [6]. A discriminator is employed to correctly

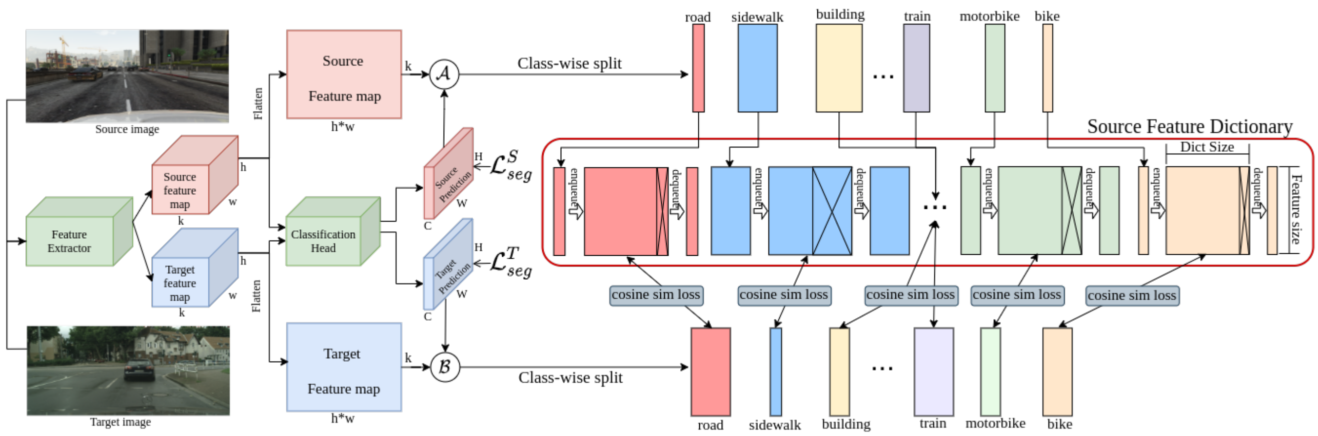

Figure 2. Overall schematic of our method. The feature extractor produces feature maps of both domains. The source and the target feature maps are flattened and split by classes via process $\mathcal{A}$ and $\mathcal{B}$ respectively. Process $\mathcal{A}$ filters source features that are correctly classified and splits them by classes. Process $\mathcal{B}$ splits the target feature map according to the pseudo-label or the prediction output. The source features are stored in the dictionary as a *queue* to maintain the latest features for each class. Then the cosine similarity loss is computed class-wise between split target features and source features stored in the dictionary.

distinguish from which domain the feature/output is generated from, while the segmentation network is trained to fool the discriminator. [21] adopts multi-level adversarial training in the output space. [23] further improves this method and shows that using an entropy map of the prediction output produces better results. Adversarial adaptation is a very common method that is widely used in domain adaptation [9, 22, 19].

Self-supervised learning (SSL), or "self-training" is a method of re-training a network with pseudo-labels of the target domain generated by a trained network that is adopted from the source domain. With the assistance of pseudo-labels, a network can be trained explicitly for the target domain as well as the source domain. [31] and [13] both utilize the pseudo-labels and propose a class-balanced and joint self-training respectively. [16] tries to minimize the intra-domain gap via self-supervised adaptation by separating the target domain into an easy and a hard split. Recently, [26] proposes a method that reconstructs the input-image from the network output to regularize the training of target domain along with self-supervised learning. Our work adopts image translation and self-supervised learning while utilizing a novel cosine similarity loss at the feature level. We do not employ the adversarial adaptation since we find it to be ineffective, which will be empirically justified in Sec.4.2 and Sec.4.4.

3. Method

In this section, we briefly talk about the loss function used for the semantic segmentation task and deeply investigate our proposed method. The schematic of our method is illustrated in Fig. 2. Our method utilizes a dictionary that stores the latest source features in a *queue*. As described in Fig. 1, we compute the class-wise cosine similarity between the target features and the source features stored in the dictionary. More detailed formulation of the proposed cosine similarity loss will be explained in Sec. 3.2. The cosine similarity loss is minimized along with the source and the target segmentation losses using the ground truth labels and the pseudo-labels respectively. Our cosine similarity loss pulls target features closer to the source features so that both domains are aligned in the feature space.

3.1. Semantic Segmentation

We follow the basic framework of unsupervised domain adaptation of semantic segmentation where there exist a labeled source domain dataset $\{x_i^s, y_i^s\}_{i=1}^{N_s}$ and a target domain dataset with only images $\{x_j^t\}_{j=1}^{N_t}$. Here, we assume that $y_i^s \in \mathbb{R}^{H\times W}$ with its elements being $y_i^{s(h,w)} \in [C]^1$. We train a segmentation network $\mathcal{G}$ that generates a prediction output $\mathcal{G}(x) = P \in \mathbb{R}^{H\times W\times C}$. We use the cross-entropy loss for the segmentation loss as follows:

$$\mathcal{L}_{seg}^{S}(x^s) = -\sum_{h,w}^{H,W}\sum_{c=1}^{C} y^{s(h,w,c)} \log(P^{s(h,w,c)}) \quad (1)$$

$$\mathcal{L}_{seg}^{T}(x^t) = -\sum_{h,w}^{H,W}\sum_{c=1}^{C} \hat{y}^{t(h,w,c)} \log(P^{t(h,w,c)}) \quad (2)$$

$$\mathcal{L}_{seg}(x^s, x^t) = \mathcal{L}_{seg}^{S}(x^s) + \mathcal{L}_{seg}^{T}(x^t) \quad (3)$$

Here H, W are height and width of the prediction output and C denotes the number of classes. The source segmen-

[1] $[C]$ is the set of natural numbers up to C. For simplicity, the notations y_i^s (label) and $\hat{y}_j^t$ (pseudo-label) are used to denote either an element of $[C]$ or a C-dimensional one-hot vector interchangeably in this paper.

tation loss is defined using the ground truth labels provided by the source dataset, on the other hand, for the target segmentation loss, we adopt self-supervised learning scheme and use pseudo-labels denoted as $\{\hat{y}^t_j\}_{j=1}^{N_t}$ which are generated from a separate trained model. Following the process of [13], only pixels with higher confidence than a threshold are filtered:

$$\begin{aligned}(C^t_{max}, P^t_{max}) &= (\arg\max_{c\in[C]} P^{t(c)}, \max_{c\in[C]} P^{t(c)}) \\ \hat{y}^t &= \mathbb{1}_{[P^t_{max} > \tau^{C^t_{max}}]} \odot C^t_{max} \in \mathbb{R}^{H\times W}\end{aligned} \tag{4}$$

where $\mathbb{1}$ is an element-wise indicator function that returns 1 if the condition is met and 0 if not. $\odot$ denotes an element-wise multiplication. If $\hat{y}^{t(h,w)} = 0$, it indicates that the pixel (h, w) is ignored. Detailed explanation for choosing the class-specific thresholds $\{\tau^c\}_{c\in[C]}$ is described in the supplementary. The overall segmentation objective is to minimize $\mathcal{L}_{seg}$ to perform pixel-level classification for both domains.

3.2. Cosine Similarity loss

The core idea of our method is to measure how similar the target features are to the source features spatial-wise and selectively maximize the similarity for certain target features that are highly similar to specific source features. First, we discuss how to split the source feature map by classes and store them to the source feature dictionary. A segmentation network, such as [1], mainly consists of two parts, a feature extractor $\mathcal{F}$ and a classification head $\mathcal{H}$, hence $\mathcal{G} = \mathcal{H} \circ \mathcal{F}$. We feed a source image $x^s \in \mathbb{R}^{H\times W\times 3}$ into $\mathcal{F}$ and generate a feature map $f^s = \mathcal{F}(x^s) \in \mathbb{R}^{h\times w\times k}$ where h, w and k represent the height, width and the feature size (number of channels) of f^s. $\mathcal{H}$ takes f^s and generates a prediction output, $p^s = \mathcal{H}(f^s) \in \mathbb{R}^{h\times w\times C}$ followed by a bilinear interpolation $P^s = I_{bilinear}(p^s) \in \mathbb{R}^{H\times W\times C}$. This can be put in one line as follows:

$$P^s = \mathcal{G}(x^s) = I_{bilinear}(\mathcal{H}(\mathcal{F}(x^s))). \tag{5}$$

We want to select correctly classified features from f^s using p^s and the ground truth label $y^s \in \mathbb{R}^{H\times W}$. We resize the ground truth label y^s into the spatial size of p^s via nearest interpolation, $\tilde{y}^s = I_{nearest}(y^s) \in \mathbb{R}^{h\times w}$.

$$\begin{aligned}c^s_{max} &= \arg\max_{c\in[C]} p^{s(c)} \in \mathbb{R}^{h\times w} \\ \hat{c}^s_{max} &= \mathbb{1}_{[c^s_{max} = \tilde{y}^s]} \odot c^s_{max} \in \mathbb{R}^{h\times w} \\ S^c &= \mathbb{1}_{[\hat{c}^s_{max} = c]} \otimes f^s.\end{aligned} \tag{6}$$

Here, $\otimes$ denotes the element-wise product of $\mathbb{1}$ and each channel slice of f^s. $\hat{c}^s_{max}$ contains information about the correctly classified output class according to p^s and has the ignore symbol (0) where it is incorrectly classified. S^c refers to the feature tensor f^s that are correctly classified as class c according to $\hat{c}^s_{max}$. Therefore, one f^s can be split into maximum C number of S^c. We flatten each S^c along the spatial dimension, meaning that it has the shape of $[k \times hw^s_c]$, where hw^s_c refers to the number of pixels in f^s that are correctly classified as c. Each S^c that corresponds to each input $\{x^s_i\}_{i=1}^{N_s}$ is enqueued into the dictionary D according to its class. D has class identities as the keys and the values of each key are source features belonging to each class. D^c refers to the values of D accessed with key c and it has the maximum size of *dict-size* which is a hyper-parameter. D is updated with new source features and the old features stored in D are dequeued at every iteration. The reason we use the dictionary is to solve the case when a class appears only in the target image and not in the source image at the current iteration, we call this as the 'unmatching problem'. In this case, the target features belonging to that class can not be matched with proper source features since the current source image does not contain the class. Also, using the dictionary allows the target features to be matched with more variety of source features from various images. D^c has the shape of $[k \times$ *dict-size*$]$ when it is fully queued. We call this process as $\mathcal{A}$.

Like the source features, we need to split a target feature map class-wise. Since target domain does not have ground truth labels, we consider two cases for splitting: when pseudo-labels are provided and when they are not. In the first case, we split using the pseudo-labels. As in (4), a pseudo-label $\hat{y}^t \in \mathbb{R}^{H\times W}$ has ignore symbols where the confidence of the trained model are lower than the class-specific threshold. Therefore, we augment $\hat{y}^t$ by replacing the ignore symbols with the prediction output of the current training network, $p^t \in \mathbb{R}^{h\times w\times C}$. We argmax p^t along the class dimension and obtain $c^t_{max} \in \mathbb{R}^{h\times w}$. $\hat{y}^t$ is resized to the spatial size of c^t_{max} as $\tilde{y}^t$ analogous to $\tilde{y}^s$.

$$\begin{aligned}c^t_{max} &= \arg\max_{c\in[C]} p^{t(c)} \in \mathbb{R}^{h\times w} \\ \dot{y}^t &= augment(\tilde{y}^t, c^t_{max}) \in \mathbb{R}^{h\times w} \\ T^c &= \mathbb{1}_{[\dot{y}^t = c]} \otimes f^t.\end{aligned} \tag{7}$$

We augment $\tilde{y}^t$ with c^t_{max}, generating $\dot{y}^t_j$ which has the values of c^t_{max} where $\tilde{y}^t$ has ignore symbols. We split a target feature map f^t according to the augmented pseudo-label $\dot{y}^t$. In the second case, we split a target feature map only according to c^t_{max}.

$$T^c = \mathbb{1}_{[c^t_{max} = c]} \otimes f^t. \tag{8}$$

Therefore, f^t can be split into maximum C number of T^c same as f^s. T^c is flattened along the spatial dimension hence its shape is $[k \times hw^t_c]$ analogous to S^c. We name this process as $\mathcal{B}$. The process $\mathcal{A}$ and $\mathcal{B}$ are illustrated in the supplementary. Next step is to compute a cosine similarity matrix between split target features and source features stored in the dictionary for each class (Also, see Fig. 3).

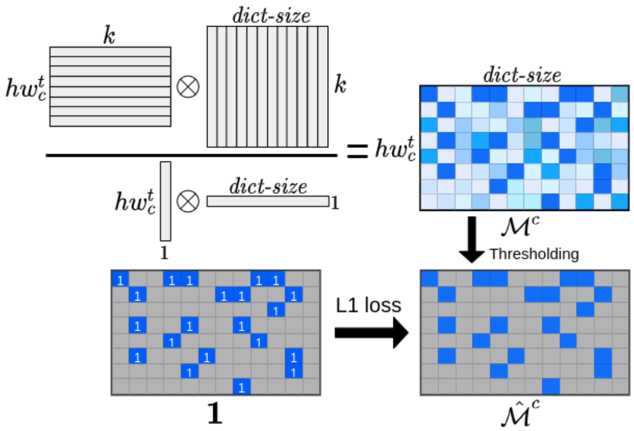

Figure 3. Dark grey colored parts refer to ignore symbols. The different blue colors for the elements of $\mathcal{M}^c$ refer to different scales of cosine similarity. Bluer means higher cosine similarity.

$$\mathcal{M}^c = \frac{T^{c\mathsf{T}} \cdot D^c}{\|T^c\|_2^{\mathsf{T}} \cdot \|D^c\|_2} \tag{9}$$

$\|T^c\|_2$ and $\|D^c\|_2$ are L2-norms of T^c and D^c with respect to the feature dimension, thus their shapes are $[1 \times hw_c^t]$ and $[1 \times$ *dict-size*$]$, assuming that D^c is fully queued. T and $\cdot$ are transpose and matrix multiplication. Therefore, $\mathcal{M}^c$ has the shape of $[hw_c^t \times$ *dict-size*$]$ which shows the cosine similarity between current target features and the source features stored in the dictionary for class c. Every element of $\mathcal{M}^c$ is a cosine similarity thus each row represents how one target feature vector is similar to every source feature vector stored in the D^c. Since cosine similarity is normalized between 1 and -1, the closer the element is to 1, the more similar the two feature vectors of target and source are. We then select elements of $\mathcal{M}^c$ that exceeds the threshold $\mathcal{T}_{cos}$ and maximize those elements to 1. We detach D^c when computing $\mathcal{M}^c$, so the loss does not back-propagate through D^c but only through T^c.

$$\begin{aligned} \hat{\mathcal{M}}^c &= \mathbb{1}_{[\mathcal{M}^c > \mathcal{T}_{cos}]} \odot \mathcal{M}^c \\ \mathcal{L}_{cos}(x^t) &= \frac{1}{C} \sum_{c=1}^{C} \left\| \hat{\mathcal{M}}^c - \mathbf{1} \right\|_1^1 \end{aligned} \tag{10}$$

$\hat{\mathcal{M}}^c$ has selected elements of $\mathcal{M}^c$ that have higher cosine similarity than $\mathcal{T}_{cos}$. $\mathbf{1}$ has the same shape as $\hat{\mathcal{M}}^c$ but filled with 1 where the elements of $\hat{\mathcal{M}}^c$ are not ignore symbols. We maximize the selected elements to 1 by minimizing L1 loss between $\hat{\mathcal{M}}^c$ and $\mathbf{1}$. Fig. 3 gives an illustration of computing the cosine similarity loss for class c. Because our method compares a target feature with every source feature in D^c, it can selectively maximize the similarity of target features to the most similar source features, making the target feature distribution and the source feature distribution aligned in the feature space.

The total loss function is to minimize the segmentation loss along with our cosine similarity loss with balance parameter λ_{cos}. (11) is used when the pseudo-labels for the target domain are **not** provided thus the target feature map is split using (8). (12) is when the pseudo-labels are available thus the target feature map is split by (7). Note that we do not employ any adversarial adaptation loss.

$$\mathcal{L}_{total}(x^s, x^t) = \mathcal{L}_{seg}^S(x^s) + \lambda_{cos}\mathcal{L}_{cos}(x^t) \tag{11}$$

$$\mathcal{L}_{total}(x^s, x^t) = \mathcal{L}_{seg}(x^s, x^t) + \lambda_{cos}\mathcal{L}_{cos}(x^t). \tag{12}$$

4. Experiments

4.1. Datasets and Training Details

Datasets. We conduct experiments on two UDA benchmarks, GTA5→Cityscapes and SYNTHIA→Cityscapes. The GTA5 [17] dataset consists of 24,966 images captured from a video game with pixel-level annotations. Originally it has annotations for 33 classes, but only 19 classes that are in common with Cityscapes are used to fairly compare with other methods. Images are resized to 1280×720 during training. The SYNTHIA [18] dataset also consists of 9,400 synthetic images with a resolution of 1280×760. Similar to GTA5, 16 common classes with Cityscapes are used for training, but for evaluation, the 16 classes and a subset with 13 classes are used following the standard protocol. The Cityscapes [5] dataset is a semantic segmentation dataset collected from real world during driving scenarios. We use 2,975 images of the train set to train the model and 500 images of the validation set to test our model, following previous works. Images are resized to 1024×512 during training.

Network Architecture and Training Details. We use two different network architectures, DeepLabV2 [1] with ResNet101 [7] backbone and FCN-8s [14] with VGG16 backbone [20]. Both networks are initialized with an ImageNet pre-trained network. We do not employ any discriminator network so the segmentation network is the only neural network in usage. Pytorch deep learning framework is used on single GPU. Batch size is set to 1 due to limited memory, same as other methods [23, 13, 28, 24]. For DeepLabV2 with ResNet101 backbone, we use SGD as the optimizer with an initial learning rate of 2.5×10^{-4} and a weight-decay of 0.0005. The learning rate is scheduled using 'poly' learning rate policy with a power of 0.9. FCN-8s with VGG16 backbone is optimized by ADAM optimizer with an initial learning rate of 1×10^{-5} and the momentums of 0.9 and 0.99. The learning rate is decayed by 'step' learning rate policy with a step size of 50,000 and a decay rate of 0.1. We adopt the source images of [13] which are transferred into the style of Cityscapes by CycleGAN [30]. Hyper-parameters used in our method such as $\mathcal{T}_{cos}$, *dict-size* and λ_{cos} will be discussed in Sec. 4.5.

Architecture	Method	GTA5→CS	SYNTHIA→CS
DeeplabV2	Adversarial	45.1%	48.5%
	Ours-(11)	46.6%	48.6%
FCN-8s	Adversarial	40.7%	36.2%
	Ours-(11)	41.8%	36.5%

Table 1. The results of training without the pseudo-labels for both tasks using two different network architectures.

4.2. Training without pseudo-labels

Tab. 1 shows our results of training without pseudo-labels. When the pseudo-labels are not provided, the adversarial adaptation loss is usually utilized to match the distribution of the target prediction to that of the source prediction. However, as mentioned earlier, adversarial adaptation is difficult to train and requires an additional discriminator network. Moreover, It is known to cause instability in the long-term training since it only sees the global prediction output and not the details. 'Adversarial' in the table refers to adversarial adaptation method [21] which tries to align the distributions of the two domains at the prediction output level using the adversarial training. Detailed formulation of it is in the supplementary. 'Ours-(11)' refers to models trained with our cosine similarity loss using (11) which splits the target feature map solely based on the target prediction output. By comparing the results of 'Ours-(11)' with 'Adversarial', we want to show that, when pseudo-labels are unavailable, our method can not only replace the adversarial adaptation but also lead to better performance results. Overall, Tab. 1 indicates that with our cosine similarity loss, adversarial adaptation loss is unnecessary and can be replaced.

4.3. Comparison with other SOTA methods

In this section, we compare our results with other state-of-the-art methods on GTA5→Cityscapes and SYNTHIA→Cityscapes using two network architectures, DeepLabV2 based on ResNet101 and FCN-8s based on VGG16. Our results are from models trained by (12). Considering that FCN-8s has a different architecture from DeepLabV2, our method is tweaked a little to adapt the difference. We basically use the cosine similarity loss for three different layers, fc7, pool4 and pool3, and more explanation is in the supplementary. We use mIoU of all 19 classes for GTA5→Cityscapes, but for SYNTHIA→Cityscapes, mIoU13 and mIoU16 are used for DeepLabV2 and FCN-8s respectively, following the standard evaluation protocol.

GTA5→Cityscapes. Tab. 2 shows our comparison experiment on GTA5→Cityscapes. For DeepLabV2-ResNet101 model, our method outperforms most of current UDA methods except for [12] and [28]. However, these two approaches are very different from ours which are to style-transfer source images into many different textures and of Cityscapes respectively. It is more about how to style-

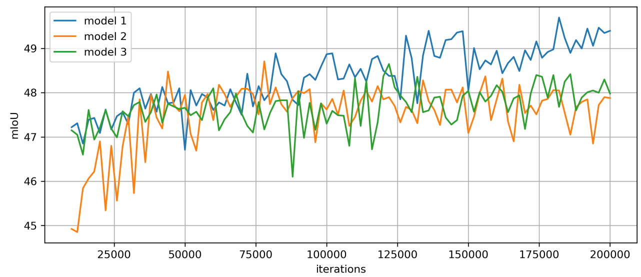

Figure 4. The ablation study on mIoU vs. iterations, model 1 - 'Ours', model 2 - 'with Adversarial', model 3 - 'only SSL'.

transfer the source images properly to adapt well rather than aligning the feature distributions of the source and the target domains. Furthermore, the mIoU of [28] is an ensemble average of three different models, not a single model, whose best single model mIoU (48.77%) is below ours. Since these two works have different directions from ours, we expect better results can be produced when our method is combined with them. For FCN-8s-VGG16 model, our method outperforms other existing methods and achieves the new state-of-the-art performance.

SYNTHIA→Cityscapes. Tab. 3 shows the results on SYNTHIA→Cityscapes. SYNTHIA→Cityscapes is a more difficult task than GTA5→Cityscapes since the domain discrepancy is much larger. The images in SYNTHIA dataset have different perspectives from Citysacpses. There are more viewpoints from a higher position such as traffic surveillance cameras. Despite the difficulty of the dataset, as can be seen in the table, our method shows significant performance in both architectures. In DeepLabV2-ResNet101 architecutre, it shows second-best performance and in FCN-8s-VGG16, it outperforms other existing methods and achieves the new state-of-the-art.

4.4. Ablation Study

In this section, we deeply investigate the contributions of our method via ablation study. By removing each building block of our work, we show its adequacy. In Tab. 4 'w/o Dictionary' refers to training without the source feature dictionary thus the cosine similarity matrix is computed only between the current iteration's source features and target features. 'w/o Class-wise Split' means without using the split process $\mathcal{A}$ and $\mathcal{B}$, thus it also does not use the dictionary and the cosine similarity matrix is computed between the current iteration's unsplit source and target features as a whole. 'with Adversarial' is to use the adversarial adaptation loss along with our $\mathcal{L}_{total}$, (12). 'only SSL' refers to training only with the segmentation loss, $\mathcal{L}_{seg}$. 'SSL with Adversarial' is a model trained by $\mathcal{L}_{seg}$ with the adversarial adaptation loss. As shown in the table, mIoU decreases when each contribution of our work is removed. The gap between 'Ours' and 'only SSL' shows the effectiveness of our cosine similarity loss. What is interesting is the results of

	Arch.	road	sidewalk	building	wall	fence	pole	light	sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorbike	bicycle	mIoU
AdvEnt[23]	R	89.4	33.1	81.0	26.6	26.8	27.2	33.5	24.7	83.9	36.7	78.8	58.7	30.5	84.8	38.5	44.5	1.7	31.6	32.4	45.5
IntraDA [16]	R	90.6	37.1	82.6	30.1	19.1	29.5	32.4	20.6	85.7	40.5	79.7	58.7	31.1	86.3	31.5	48.3	0.0	30.2	35.8	46.3
PLCA [11]	R	84.0	30.4	82.4	35.3	24.8	32.2	36.8	24.5	85.5	37.2	78.6	66.9	32.8	85.5	40.4	48.0	8.8	29.8	41.8	47.7
BDL [13]	R	91.0	44.7	84.2	34.6	27.6	30.2	36.0	36.0	85.0	43.6	83.0	58.6	31.6	83.3	35.3	49.7	3.3	28.8	35.6	48.5
CrCDA [10]	R	92.4	55.3	82.3	31.2	29.1	32.5	33.2	35.6	83.5	34.8	84.2	58.9	32.2	84.7	40.6	46.1	2.1	31.1	32.7	48.6
SIM [24]	R	90.6	44.7	84.8	34.3	28.7	31.6	35.0	37.6	84.7	43.3	85.3	57.0	31.5	83.8	42.6	48.5	1.9	30.4	39.0	49.2
Label-driven[26]	R	90.8	41.4	84.7	35.1	27.5	31.2	38.0	32.8	85.6	42.1	84.9	59.6	34.4	85.0	42.8	52.7	3.4	30.9	38.1	49.5
Kim et al. [12]	R	92.9	55.0	85.3	34.2	31.1	34.9	40.7	34.0	85.2	40.1	87.1	61.0	31.1	82.5	32.3	42.9	0.3	36.4	46.1	50.2
FDA-MBT [28]	R	92.5	53.3	82.4	26.5	27.6	36.4	40.6	38.9	82.3	39.8	78.0	62.6	34.4	84.9	34.1	53.1	16.9	27.7	46.4	**50.45**
Ours	R	92.6	54.0	85.4	35.0	26.0	32.4	41.2	29.7	85.1	40.9	85.4	62.6	34.7	85.7	35.6	50.8	2.4	31.0	34.0	49.7
CrDoCo [4]	V	89.1	33.2	80.1	26.9	25.0	18.3	23.4	12.8	77.0	29.1	72.4	55.1	20.2	79.9	22.3	19.5	1.0	20.1	18.7	38.1
CrCDA [10]	V	86.8	37.5	80.4	30.7	18.1	26.8	25.3	15.1	81.5	30.9	72.1	52.8	19.0	82.1	25.4	29.2	10.1	15.8	3.7	39.1
BDL [13]	V	89.2	40.9	81.2	29.1	19.2	14.2	29.0	19.6	83.7	35.9	80.7	54.7	23.3	82.7	25.8	28.0	2.3	25.7	19.9	41.3
FDA-MBT [28]	V	86.1	35.1	80.6	30.8	20.4	27.5	30.0	26.0	82.1	30.3	73.6	52.5	21.7	81.7	24.0	30.5	29.9	14.6	24.0	42.2
Kim et al. [12]	V	92.5	54.5	83.9	34.5	25.5	31.0	30.4	18.0	84.1	39.6	83.9	53.6	19.3	81.7	21.1	13.6	17.7	12.3	6.5	42.3
SIM [24]	V	88.1	35.8	83.1	25.8	23.9	29.2	28.8	28.6	83.0	36.7	82.3	53.7	22.8	82.3	26.4	38.6	0.0	19.6	17.1	42.4
Label-driven[26]	V	90.1	41.2	82.2	30.3	21.3	18.3	33.5	23.0	84.1	37.5	81.4	54.2	24.3	83.0	27.6	32.0	8.1	29.7	26.9	43.6
Ours	V	90.3	42.6	82.2	29.7	22.2	18.5	32.8	26.8	84.3	37.1	80.2	55.2	26.4	83.0	30.3	35.1	7.0	29.6	28.9	**44.3**

Table 2. Performance comparison of our method with other state-of-the-art methods on **GTA5→Cityscapes**. 'R' and 'V' refer to DeepLabV2-ResNet101 and FCN-8s-VGG16 respectively.

	Arch.	road	sidewalk	building	wall	fence	pole	light	sign	vegetation	sky	person	rider	car	bus	motorbike	bicycle	mIoU
AdvEnt[23]	R	85.6	42.2	79.7	—	—	—	5.4	8.1	80.4	84.1	57.9	23.8	73.3	36.4	14.2	33.0	48.0
IntraDA [16]	R	84.3	37.7	79.5	—	—	—	9.2	8.4	80.0	84.1	57.2	23.0	78.0	38.1	20.3	36.5	48.9
Kim et al. [12]	R	92.6	53.2	79.2	—	—	—	1.6	7.5	78.6	84.4	52.6	20.0	82.1	34.8	14.6	39.4	49.3
CrCDA [10]	R	86.2	44.9	79.5	—	—	—	9.4	11.8	78.6	86.5	57.2	26.1	76.8	39.9	21.5	32.1	50.0
BDL [13]	R	86.0	46.7	80.3	—	—	—	14.1	11.6	79.2	81.3	54.1	27.9	73.7	42.2	25.7	45.3	51.4
SIM [24]	R	83.0	44.0	80.3	—	—	—	17.1	15.8	80.5	81.8	59.9	33.1	70.2	37.3	28.5	45.8	52.1
FDA-MBT [28]	R	79.3	35.0	73.2	—	—	—	19.9	24.0	61.7	82.6	61.4	31.1	83.9	40.8	38.4	51.1	52.5
Label-driven[26]	R	85.1	44.5	81.0	—	—	—	16.4	15.2	80.1	84.8	59.4	31.9	73.2	41.0	32.6	44.7	53.1
PLCA[11]	R	82.6	29.0	81.0	—	—	—	24.9	18.3	82.8	82.3	62.1	26.5	85.6	48.9	26.8	52.2	**54.0**
Ours	R	88.3	47.3	80.1	—	—	—	21.6	20.2	79.6	82.1	59.0	28.2	82.0	39.2	17.3	46.7	53.2
CrCDA [10]	V	74.5	30.5	78.6	6.6	0.7	21.2	2.3	8.4	77.4	79.1	45.9	16.5	73.1	24.1	9.6	14.2	35.2
ROAD-Net [3]	V	77.7	30.0	77.5	9.6	0.3	25.8	10.3	15.6	77.6	79.8	44.5	16.6	67.8	14.5	7.0	23.8	36.2
GIO-Ada [2]	V	78.3	29.2	76.9	11.4	0.3	26.5	10.8	17.2	81.7	81.9	45.8	15.4	68.0	15.9	7.5	30.4	37.3
CrDoCo [4]	V	84.9	32.8	80.1	4.3	0.4	29.4	14.2	21.0	79.2	78.3	50.2	15.9	69.8	23.4	11.0	15.6	38.2
BDL [13]	V	72.0	30.3	74.5	0.1	0.3	24.6	10.2	25.2	80.5	80.0	54.7	23.2	72.7	24.0	7.5	44.9	39.0
FDA-MBT [28]	V	84.2	35.1	78.0	6.1	0.44	27.0	8.5	22.1	77.2	79.6	55.5	19.9	74.8	24.9	14.3	40.7	40.5
Label-driven[26]	V	73.7	29.6	77.6	1.0	0.4	26.0	14.7	26.6	80.6	81.8	57.2	24.5	76.1	27.6	13.6	46.6	41.1
Ours	V	73.6	30.6	77.5	0.8	0.4	26.7	14.1	29.3	80.9	80.6	57.9	24.7	76.5	27.2	10.8	47.8	**41.2**

Table 3. Performance comparison of our method with other state-of-the-art methods on **SYNTHIA→Cityscapes**. 'R' and 'V' refer to DeepLabV2-ResNet101 and FCN-8s-VGG16 respectively. mIoU13 and mIoU16 are used for 'R' and 'V'.

Arch.	Method	GTA5→CS	SYNTHIA→CS
R	Ours (full model)	49.7%	53.2%
	w/o Dictionary	49.33%	52.64%
	w/o Class-wise Split	48.93%	52.67%
	with Adversarial	48.71%	52.75%
	only SSL	48.65%	52.48%
	SSL with Adversarial	48.41%	52.56%
V	Ours (full model)	44.3%	41,2%
	only SSL	43.52%	40.81%

Table 4. The ablation study results. Note that 'R' and 'V' refer to DeepLabV2-ResNet101 and FCN-8s-VGG16 respectively.

'with Adversarial': the mIoU rather decreases which means the adversarial adaptation loss disturbs the training. The gap between 'only SSL' and 'SSL with Adversarial' also supports this. For SYNTHIA→Cityscapes, which has a larger domain gap, adversarial adaptation leads to marginal performance gain but for GTA5→Cityscapes, the result is rather worsen.

This can be more clearly observed in Fig. 4. It shows the plots of mIoU on validation set at every 2,000 iteration during training. Model 1, 2 and 3 refer to 'Ours', 'with Adversarial' and 'only SSL' models (DeepLabV2-ResNet101) trained on GTA5→Cityscapes respectively. The plots show a clear gap between 'Ours' and other two models. The mIoU of model 2 increases fast until certain iteration but stops gaining any additional performance since then. Moreover, its plot fluctuates excessively which shows the instability of adversarial training. The clear gap between model 1 and model 3 shows the validity of our cosine similarity loss. Also, 'Ours' constantly moves upwards as the iteration goes on, showing the possibility of further improvement in further iterations.

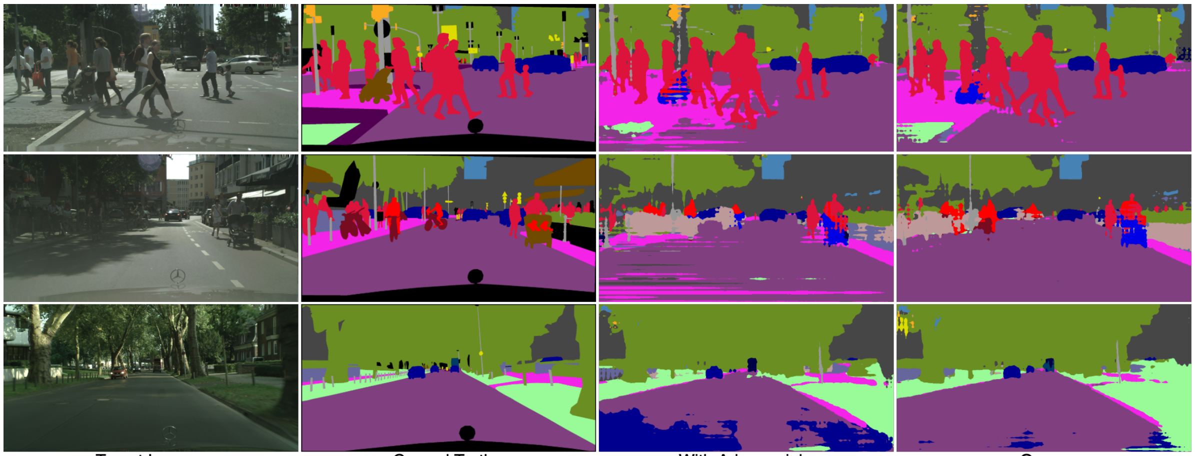

Figure 5. Qualitative Comparison. From left to right, it shows the target image, ground truth, prediction outputs of 'with Adversarial' and 'Ours'. 'Ours' shows much clear prediction outputs than 'with Adversarial'.

Task	Arch.	$\mathcal{T}_{cos}$	*dict-size*	λ_{cos}
GTA5→CS	R	0.6	2500	0.01
	V	0.4	2000	0.001
SYNTHIA→CS	R	0.4	2500	0.01
	V	0.2	2000	0.001

Table 5. Hyper-parameters used for our method according to the task and the architecture.

$\mathcal{T}_{cos}$	0.5	0.55	0.6	0.65	0.7
mIoU	48.92%	49.15%	**49.7%**	49.06%	49.47%
dict-size	1000	1500	2000	2500	3000
mIoU	49.07%	49.51%	49.26%	**49.7%**	49.13%

Table 6. Hyper-parameters analysis. *dict-size* is set to 2500 while analyzing $\mathcal{T}_{cos}$ and $\mathcal{T}_{cos}$ is set to 0.6 while analyzing *dict-size*.

4.5. Hyper-parameter Analysis

There are three hyper-parameters used in our cosine similarity loss which are $\mathcal{T}_{cos}$, *dict-size* and λ_{cos}. $\mathcal{T}_{cos}$ determines the amount of target features to be maximized. If it is set too low, almost every target features would be selected and possibly be maximized with even dissimilar source features. If it is set too high, not enough target features would be selected thus the cosine similarity loss would have little effect. *dict-size* decides how many source feature vectors are to be stored in the dictionary for each class. The larger its size, the older the features that can be stored. λ_{cos} is simply used to balance the loss. $\mathcal{T}_{cos}$ and *dict-size* are important hyper-parameters of our cosine similarity loss, hence the values are different by architecture, dataset and pseudo labels, but we find that the hyper-parameters shown in Tab. 5 work in general. Tab. 6 shows our hyper-parameter analysis on GTA5→Cityscapes using DeepLabV2.

4.6. Qualitative Comparison

Fig. 5 shows qualitative comparison results. The first two columns are target images and the corresponding ground truth labels while the last two columns are qualitative results of "with Adversarial" and "Ours". As can be seen from the figure, when the adversarial adaptation loss is additionally applied, the prediction outputs are more noisy and not clear. It incorrectly classifies the road as some other classes. 'Ours' has much smoother and more clear prediction outputs. Furthermore, our method performs better at recognizing distant objects, for example, people on the right side of the second row. We conjecture that this is due to our cosine similarity loss that tries to adapt target features to the specific source features that are most similar.

5. Conclusion

We propose maximizing cosine similarity between the source and the target domain at the feature level to tackle the problem of unsupervised domain adaptation for semantic segmentation. Our method measures the cosine similarity between a target feature and every source features spatially by classes and selectively maximizes the similarity only with the most semantically similar source features. We also propose source feature dictionary to maintain the latest source features which enables target features to be maximized with various source features and prevents the unmatching problem. We empirically show that our method can replace the unstable adversarial adaptation which is incapable of selectively adapting the target features to the most related source features. We hope our work can be further studied and developed in the future work.

6. Acknowledgments

This work was supported by NRF grant (2021R1A2C3006659) and IITP grant (No. 2021-0-01343,Artificial Intelligence Graduate School Program), both funded by Korean Government.

References

[1] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern@inproceedingsyang2021context, title=Context-aware domain adaptation in semantic segmentation, author=Yang, Jinyu and An, Weizhi and Yan, Chaochao and Zhao, Peilin and Huang, Junzhou, booktitle=Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pages=514–524, year=2021 analysis and machine intelligence*, 40(4):834–848, 2017.

[2] Yuhua Chen, Wen Li, Xiaoran Chen, and Luc Van Gool. Learning semantic segmentation from synthetic data: A geometrically guided input-output adaptation approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1841–1850, 2019.

[3] Yuhua Chen, Wen Li, and Luc Van Gool. Road: Reality oriented adaptation for semantic segmentation of urban scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7892–7901, 2018.

[4] Yun-Chun Chen, Yen-Yu Lin, Ming-Hsuan Yang, and Jia-Bin Huang. Crdoco: Pixel-level domain transfer with cross-domain consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1791–1800, 2019.

[5] 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. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27:2672–2680, 2014.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[8] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *International conference on machine learning*, pages 1989–1998. PMLR, 2018.

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

[10] Jiaxing Huang, Shijian Lu, Dayan Guan, and Xiaobing Zhang. Contextual-relation consistent domain adaptation for semantic segmentation. In *European Conference on Computer Vision*, pages 705–722. Springer, 2020.

[11] Guoliang Kang, Yunchao Wei, Yi Yang, Yueting Zhuang, and Alexander G Hauptmann. Pixel-level cycle association: A new perspective for domain adaptive semantic segmentation. *arXiv preprint arXiv:2011.00147*, 2020.

[12] Myeongjin Kim and Hyeran Byun. Learning texture invariant representation for domain adaptation of semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12975–12984, 2020.

[13] Yunsheng Li, Lu Yuan, and Nuno Vasconcelos. Bidirectional learning for domain adaptation of semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6936–6945, 2019.

[14] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[15] Luigi Musto and Andrea Zinelli. Semantically adaptive image-to-image translation for domain adaptation of semantic segmentation. *arXiv preprint arXiv:2009.01166*, 2020.

[16] Fei Pan, Inkyu Shin, Francois Rameau, Seokju Lee, and In So Kweon. Unsupervised intra-domain adaptation for semantic segmentation through self-supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3764–3773, 2020.

[17] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *European Conference on Computer Vision (ECCV)*, volume 9906 of *LNCS*, pages 102–118. Springer International Publishing, 2016.

[18] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio Lopez. The SYNTHIA Dataset: A large collection of synthetic images for semantic segmentation of urban scenes. 2016.

[19] Rui Shu, Hung H Bui, Hirokazu Narui, and Stefano Ermon. A dirt-t approach to unsupervised domain adaptation. *arXiv preprint arXiv:1802.08735*, 2018.

[20] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[21] Yi-Hsuan Tsai, Wei-Chih Hung, Samuel Schulter, Kihyuk Sohn, Ming-Hsuan Yang, and Manmohan Chandraker. Learning to adapt structured output space for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7472–7481, 2018.

[22] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7167–7176, 2017.

[23] Tuan-Hung Vu, Himalaya Jain, Maxime Bucher, Matthieu Cord, and Patrick Pérez. Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2517–2526, 2019.

[24] Zhonghao Wang, Mo Yu, Yunchao Wei, Rogerio Feris, Jinjun Xiong, Wen-mei Hwu, Thomas S Huang, and Honghui Shi. Differential treatment for stuff and things: A simple unsupervised domain adaptation method for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Com-*

puter Vision and Pattern Recognition, pages 12635–12644, 2020.

[25] 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 *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 518–534, 2018.

[26] Jinyu Yang, Weizhi An, Sheng Wang, Xinliang Zhu, Chaochao Yan, and Junzhou Huang. Label-driven reconstruction for domain adaptation in semantic segmentation. *arXiv preprint arXiv:2003.04614*, 2020.

[27] Jinyu Yang, Weizhi An, Chaochao Yan, Peilin Zhao, and Junzhou Huang. Context-aware domain adaptation in semantic segmentation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 514–524, 2021.

[28] Yanchao Yang and Stefano Soatto. Fda: Fourier domain adaptation for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4085–4095, 2020.

[29] Lei Zhang. Transfer adaptation learning: A decade survey. *arXiv preprint arXiv:1903.04687*, 2019.

[30] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

[31] Yang Zou, Zhiding Yu, BVK Kumar, and Jinsong Wang. Domain adaptation for semantic segmentation via class-balanced self-training. *arXiv preprint arXiv:1810.07911*, 2018.