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

A. How to generate the pseudo-labels

We basically follow the process of generating pseudo-labels proposed by [1]. We assume a segmentation network $\mathcal{G}$ is already trained. As mentioned in the main paper, a segmentation network generates a prediction output $\mathcal{G}(x^t) = P^t \in \mathbb{R}^{H\times W\times C}$. By following the equation below, we generate a pseudo-label $\hat{y}^t$ that corresponds to an input image x^t.

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

We argmax along class dimension and filter only pixels whose prediction confidence exceed a class-specific confidence threshold τ^c. τ^c is the confidence threshold of class c. τ^c is set by the confidence score of top 50% of each class. We infer all the images in the training set into the network to obtain the prediction output for each image. Then, for each class, we collect all the prediction pixels that are classified as the class and add the confidence score of each pixel to a list. We sort the list in a descending order and choose the median value of the list as the τ^c for the class. If the median value is higher than 0.9, we set τ^c as 0.9. Therefore, each class has its own τ^c and it can be different by classes.

B. Figure of process $\mathcal{A}$ and $\mathcal{B}$

Fig. B.1 illustrates our process $\mathcal{A}$ and $\mathcal{B}$. The illustration of process $\mathcal{B}$ corresponds to the case when the pseudo-label is available. Process $\mathcal{A}$ first selects features that are correctly classified using $\hat{c}^s_{max}$. $\hat{c}^s_{max}$ is the correctly classified prediction output classes of c^s_{max} via the ground truth label $\tilde{y}^s$. The grey shaded parts of $\hat{c}^s_{max}$ and f^s are incorrectly classified prediction outputs and source features respectively. Then, we split the source features f^s into classes according to $\hat{c}^s_{max}$. The split source features are enqueued to the dictionary class by class.

Process $\mathcal{B}$ first augments the pseudo-label $\tilde{y}^t$ with the target prediction output. As it can be seen in the figure, a pseudo-label $\tilde{y}^t$ has ignore symbols where the confidence score of the separate trained network is lower than the confidence threshold τ^c. We augment $\tilde{y}^t$ by replacing those ignore symbols with the prediction output classes of the current training network, c^t_{max}. $\dot{y}^t$ is the augmented pseudo-label and it is basically a copy of $\tilde{y}^t$ but has the values of c^t_{max} where $\tilde{y}^t$ has ignore symbols. Then, we split the target feature map f^t class-wise according to $\dot{y}^t$.

C. Formulation of Adversarial Adaptation

Along with the segmentation loss, usually an adversarial adaptation loss is adopted to make the distribution of P_t closer to P_s by fooling a discriminator network $\mathcal{D}$. It tries to maximize the probability of target predictions being considered as source predictions by $\mathcal{D}$.

$$\mathcal{L}_{adv}(x^t) = -\sum_{h,w} \log(\mathcal{D}(P^t)^{(h,w)}) \tag{2}$$

On the other hand, $\mathcal{D}$ is trained to correctly distinguish the originating domain of the segmentation output.

$$\begin{aligned}\mathcal{L}_D(x^s, x^t) = -\sum_{h,w} &(\log(1 - \mathcal{D}(P^t)^{(h,w)}) \\ &+ \log(\mathcal{D}(P^s)^{(h,w)}))\end{aligned} \tag{3}$$

The discriminator network is a fully convolutional network which consists of 5 convolutional layers with 4×4 kernels and a stride of 2, the channel sizes are set as {64, 128, 256, 512, 1} for each layer respectively. The first four layers are followed by leaky ReLU parameterized by 0.2. We use ADAM optimizer with learning a rate of 1×10^{-4} for DeepLabV2 based on ResNet101 and 1×10^{-6} for FCN-8s based on VGG16. The momentums are set as 0.9 and 0.99. The loss balance parameter λ_{adv} is set as 0.001 and 0.0001 for DeepLabV2 and FCN-8s respectively. However, in our method, we do not employ any adversarial loss since we find it ineffective for self-supervised learning and moreover

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

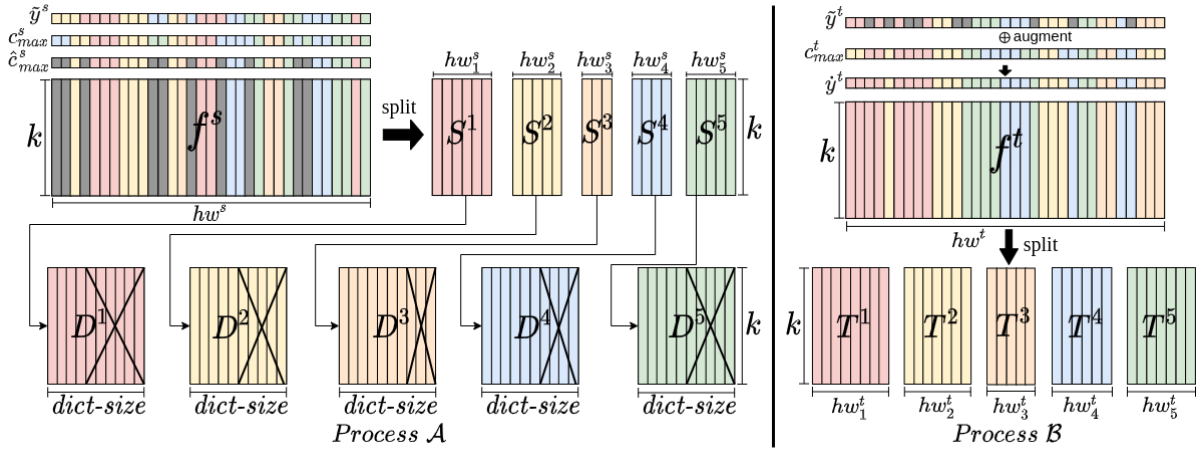

Figure B.1. Grey shaded parts in process $\mathcal{A}$ are incorrectly classified source prediction outputs and features, while in $\mathcal{B}$, they are ignore symbols of the pseudo-label. Note that $\tilde{y}_j^t$ is augmented by replacing the ignore symbols with the predicted classes of the target output, c_{max}^t, generating $\dot{y}^t$.

it disrupts the training when combined with our proposed cosine similarity loss.

D. Tweaked method for FCN

FCN-8s has a different architecture from DeepLabV2. The main difference is that it does not use the bilinear interpolation but instead uses skip combining to fuse the outputs from shallow layers and a transposed convolutional layer for upsampling. It combines the outputs from three different layers which are fc7, pool4 and pool3. Therefore, our cosine similarity loss is applied to these three different layers. We name the three different feature maps from the three layers as $\{f_l^s\}_{l=1}^3$. Since FCN-8s does not use the bilinear interpolation but rather directly produces the prediction output, we have to downsample the prediction output $P^s = \mathcal{G}(x^s) \in \mathbb{R}^{H\times W\times C}$ into three different spatial sizes of the feature maps $\{f_l^s\}_{l=1}^3$. We first argmax P^s along the class dimension and obtain predicted class information.

$$C_{max}^s = \underset{c\in[C]}{\arg\max}\, P^{s(c)} \in \mathbb{R}^{H\times W} \tag{4}$$

Then we resize C_{max}^s into the three spatial sizes of $\{f_l^s\}_{l=1}^3$ via nearest interpolation. $c_{max}^s = I_{nearest}(C_{max}^s)$ The three resized class outputs are $\{c_{max}^s{}_l\}_{l=1}^3$ which have the same spatial sizes as $\{f_l^s\}_{l=1}^3$ respectively. We also resize the ground-truth label y^s to the spatial size of each $\{c_{max}^s{}_l\}_{l=1}^3$ thus generate three resized ground truth labels $\{\tilde{y}_l^s\}_{l=1}^3$. We use $\{c_{max}^s{}_l\}_{l=1}^3$ and $\{\tilde{y}_l^s\}_{l=1}^3$ to split the source feature maps analogous to (6) of the main paper.

$$\begin{aligned} \hat{c}_{max}^s{}_l &= \mathbb{1}_{[c_{max}^s{}_l=\tilde{y}_l^s]} \odot c_{max}^s{}_l \\ S_l^c &= \mathbb{1}_{[\hat{c}_{max}^s{}_l=c]} \otimes f_l^s. \end{aligned} \tag{5}$$

S_l^c refers to the split source features from layer l that is correctly classified as class c. There are three different dictionaries each corresponding to each layer, $\{D_l\}_{l=1}^3$. Each S_l^c is enqueued to D_l^c.

This process is analogously applied to the target feature map as well. We resize the target prediction output P^t to the spatial size of the three target feature maps $\{f_l^t\}_{l=1}^3$, generating $\{c_{max}^t{}_l\}_{l=1}^3$. The pseudo-label $\hat{y}^t$ is also resized to the spatial size of each $\{c_{max}^t{}_l\}_{l=1}^3$ as $\{\tilde{y}_l^t\}_{l=1}^3$.

$$\begin{aligned} \dot{y}_l^t &= augment(\tilde{y}_l^t, c_{max}^t{}_l) \\ T_l^c &= \mathbb{1}_{[\dot{y}_l^t=c]} \otimes f_l^t. \end{aligned} \tag{6}$$

T_l^c refers to split target features from layer l that is classified as class c.

$$\begin{aligned} \mathcal{M}_l^c &= \frac{{T_l^c}^\mathsf{T} \cdot D_l^c}{\|T_l^c\|_2^\mathsf{T} \cdot \|D_l^c\|_2} \\ \hat{\mathcal{M}}_l^c &= \mathbb{1}_{[\mathcal{M}_l^c > \mathcal{T}_{cosl}]} \odot \mathcal{M}_l^c. \end{aligned} \tag{7}$$

The cosine matrix is computed for each class of each layer. $\mathcal{M}_l^c$ is the cosine matrix of class c between source features stored in D_l^c and split target features from layer l classified as class c. We only select elements of $\mathcal{M}_l^c$ that exceed $\mathcal{T}_{cosl}$. $\mathcal{T}_{cos}$ is defined differently for each layer l. A higher threshold is defined for a shallower layer since the shallow layer possesses more global information without the detailed information while features from a higher layer posses high-level abstraction that is more detailed. Therefore, we want to set a higher threshold for a shallow layer in order to maximize the similarity of the target features with more meaningful and similar source features. We simply add 0.1 to the baseline threshold as the layer goes shallower, for example, if we set 0.5 as the baseline threshold, the $\mathcal{T}_{cosl}$ for fc7, pool4 and pool3 layers are set as 0.5, 0.6, 0.7. The final

cosine similarity loss is averaged over the three layers.

$$\mathcal{L}_{cos}(x^t) = \frac{1}{3 \cdot C} \sum_{l=1}^{3} \sum_{c=1}^{C} \left\| \hat{\mathcal{M}}_l^c - \mathbf{1} \right\|_1^1. \tag{8}$$

E. More qualitative results

Fig. E.1 shows more qualitative comparison results between "with Adversarial" and "Ours". As already mentioned in the main paper, ours shows much clear and less noisy outputs.

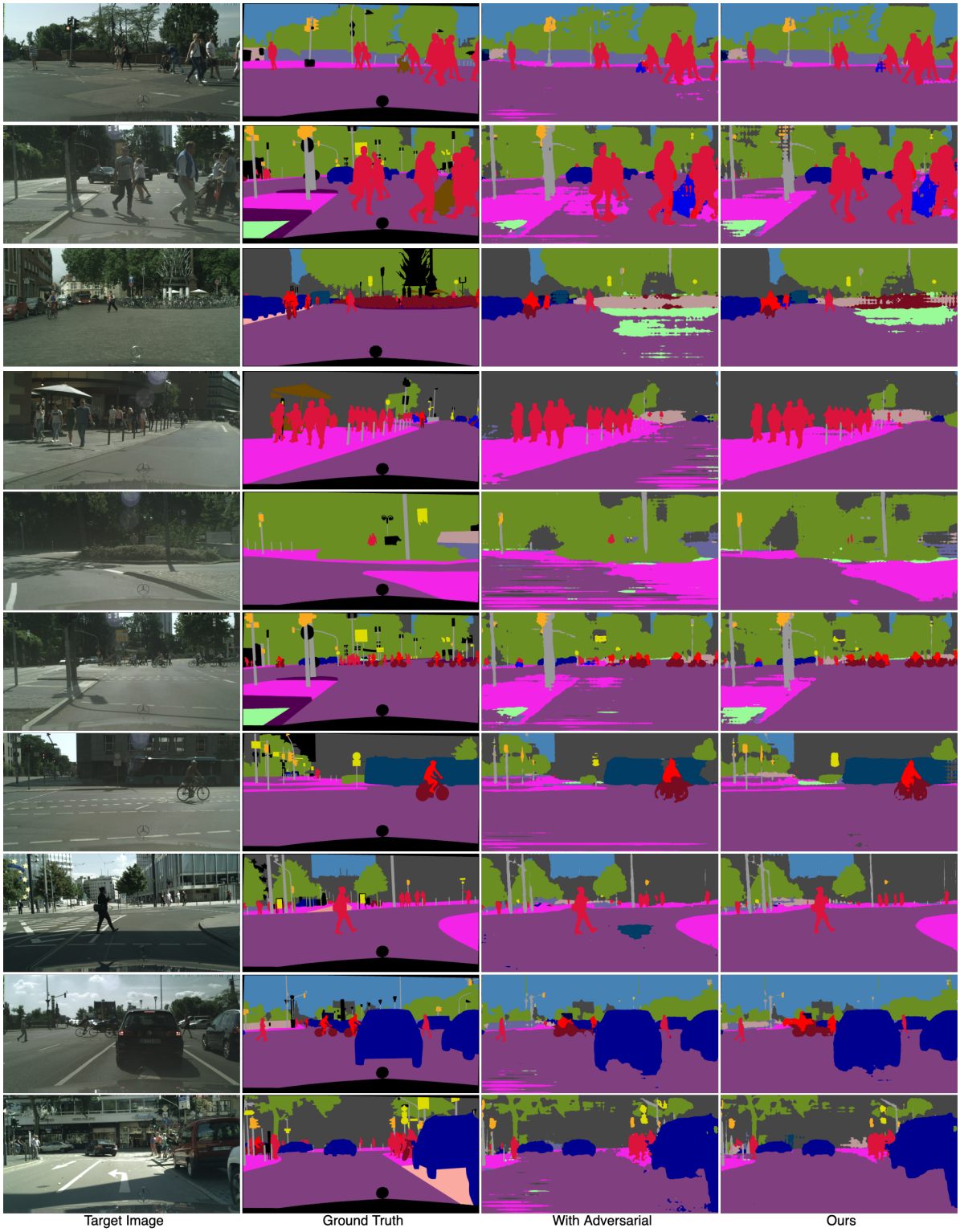

Figure E.1. More qualitative results.

References

[1] 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.