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

Inseop Chung\textsuperscript{1*} Daesik Kim\textsuperscript{2} Nojun Kwak\textsuperscript{1}
\textsuperscript{1}Seoul National University \textsuperscript{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 \cite{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 $\tilde{y}^t$ that corresponds to an input image $x^t$.

\begin{equation}
(C_{\text{max}}^t, P_{\text{max}}^t) = (\arg\max_{c \in [C]} P^{t(c)}), \max_{c \in [C]} P^{t(c)}
\end{equation}

\begin{equation}
\tilde{y}^t = \mathbb{1}_{[P_{\text{max}}^t > \tau_c]} \odot C_{\text{max}}^t \in \mathbb{R}^{H \times W}
\end{equation}

We argmax along class dimension and filter only pixels whose prediction confidence exceed a class-specific confidence threshold $\tau^c$. $\tau^c$ is the confidence threshold of class $c$. $\tau^c$ 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 $\tau^c$ for the class. If the median value is higher than 0.9, we set $\tau^c$ as 0.9. Therefore, each class has its own $\tau^c$ and it can be different by classes.

B. Figure of process $A$ and $B$

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

\* This work is conducted when the author was in research internship at NAVER WEBTOON Corp.
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 \( \{ \mathcal{f}_l \}_{l=1}^{3} \). Since FCN-8s does not use the bilinear interpolation but rather directly produces the prediction output, we have to downsample the prediction output \( \mathcal{P}_s = \mathcal{G}(x^s) \in \mathbb{R}^{H \times W \times C} \) into three different spatial sizes of the feature maps \( \{ \mathcal{f}_l \}_{l=1}^{3} \). We first argmax \( \mathcal{P}_s \) along the class dimension and obtain predicted class information.

\[
C_{\text{max}}^s = \arg \max_{c \in [C]} P_s(c) \in \mathbb{R}^{H \times W} \tag{4}
\]

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

\[
\begin{align*}
\hat{c}_{\text{max}}^l &= 1_{[c_{\text{max}}^s = \hat{y}^l]} \odot C_{\text{max}}^l \\
S_l^c &= 1_{[c_{\text{max}}^l = c]} \odot \mathcal{f}_l^c.
\end{align*} \tag{5}
\]

\( S_l^c \) 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 \( \mathcal{P}_t \) to the spatial size of the three target feature maps \( \{ \mathcal{f}_l^t \}_{l=1}^{3} \), generating \( \{ c_{\text{max}}^l \}_{l=1}^{3} \). The pseudo-label \( \hat{y}^t \) is also resized to the spatial size of each \( c_{\text{max}}^l \) as \( \{ \hat{y}^l_t \}_{l=1}^{3} \).

\[
\hat{y}^t_l = \text{augment}(\hat{y}^l_t, c_{\text{max}}^l) \tag{6}
\]

\[
T_l^c = 1_{[y^t_l = c]} \odot \mathcal{f}_l^c.
\]

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

\[
\begin{align*}
\mathcal{M}_l^c &= \frac{T_l^c \cdot D_l^c}{||T_l^c||_2 \cdot ||D_l^c||_2} \tag{7}
\end{align*}
\]

\[
\tilde{\mathcal{M}}_l^c = 1_{[c_{\text{max}}^l > \text{Thresh}]} \odot \mathcal{M}_l^c.
\]

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 \( \text{Thresh} \). \( \text{Thresh} \) 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 possesses high-level abstraction that is more detailed. Therefore, we want to set a higher threshold for a shallower 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 \( \text{Thresh} \) for fc7, pool4 and pool3 layers are set as 0.5, 0.6, 0.7. The final

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 \( \tilde{y}^t \) is augmented by replacing the ignore symbols with the predicted classes of the target output, \( c_{\text{max}}^l \), generating \( \hat{y}^t \).
cosine similarity loss is averaged over the three layers.

\[
\mathcal{L}_{\text{cos}}(x^t) = \frac{1}{3 \cdot C} \sum_{t=1}^{3} \sum_{c=1}^{C} \| \mathcal{M}_{c} - 1 \|_1.
\]  
(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