DecoupleNet: Decoupled Network for Domain Adaptive Semantic Segmentation — Supplementary Material

Introduction

This is the supplementary material, which is divided into the following sections.

1. Pseudo code is given in Sec. 1.
2. Experimental results for VGG-16 are shown in Sec. 2.
3. Experimental results on Cross-City are shown in Table. 3.
4. Multi-round performance comparison with ProDA is given in Sec. 3.
5. Implementation detail of class-wise thresholds in SD is explained in Sec. 4.
6. Analysis of extra training computational cost is shown in Sec. 5.
7. Details of the plot in Fig. 1(c) of the submission file are given in Sec. 6.
8. Visual comparison with existing methods is shown in Sec. 7.
9. Extension to UDA classification task is shown in Sec. 8.
10. More implementation details of the experiments are explained in Sec. 9.
11. t-SNE visualizations and the details are given in Sec. 10.

1 Pseudo Code

In order to clarify the back-propagation routine of the gradients, we demonstrate the pseudo code of the first-training stage (with DecoupleNet and Self-Discrimination), as shown in Alg. 1. It mainly shows which modules are updated according to each loss.

2 Experimental Results for VGG-16

Table 1 and Table 2 show the results with VGG-16 [12] on GTA5→Cityscapes and Synthia→Cityscapes, respectively. Our approach outperforms others by a large margin, which clearly demonstrates the superiority of our method.

3 Multi-round Comparison with ProDA

We compare multi-round performance of ProDA and ours in Table 4. Our method still yields a considerable performance boost by applying one more self-training round. Note that one round refers to re-labeling the target domain images based on the model in the previous round and then re-training.

Algorithm 1 First-stage training (DecoupleNet and Self-Discrimination)

Require: x_s, x_t, y_s, $\boldsymbol{W}_{\boldsymbol{g}_{src}}$, $\boldsymbol{W}_{\boldsymbol{g}_{tgt}}$, $\boldsymbol{W}_{\boldsymbol{g}_{share}}$, $\boldsymbol{W}_{\mathcal{C}}$, $\boldsymbol{W}_{\mathcal{C}_{aux}}$, $\boldsymbol{W}_{\mathcal{D}_{low}}$, $\boldsymbol{W}_{\mathcal{D}}$

for $iter = 1 \rightarrow num\_iters$ **do**
  $\nabla \boldsymbol{W}_{\boldsymbol{g}_{src}} \leftarrow 0$, $\nabla \boldsymbol{W}_{\boldsymbol{g}_{tgt}} \leftarrow 0$, $\nabla \boldsymbol{W}_{\boldsymbol{g}_{share}} \leftarrow 0$, $\nabla \boldsymbol{W}_{\boldsymbol{C}} \leftarrow 0$, $\nabla \boldsymbol{W}_{\boldsymbol{C}_{aux}} \leftarrow 0$

  Obtain $\mathcal{L}_{ce}$ according to Eq. (8)
  $\nabla \boldsymbol{W}_{\boldsymbol{g}_{src}} \leftarrow \frac{\partial \mathcal{L}_{ce}}{\partial \boldsymbol{W}_{\boldsymbol{g}_{src}}}$
  $\nabla \boldsymbol{W}_{\boldsymbol{g}_{share}} \leftarrow \frac{\partial \mathcal{L}_{ce}}{\partial \boldsymbol{W}_{\boldsymbol{g}_{share}}}$
  $\nabla \boldsymbol{W}_{\mathcal{C}} \leftarrow \frac{\partial \mathcal{L}_{ce}}{\partial \boldsymbol{W}_{\mathcal{C}}}$

  Obtain $\mathcal{L}_{adv}^{low}$ according to Eq. (9)
  $\nabla \boldsymbol{W}_{\boldsymbol{g}_{src}} \leftarrow \nabla \boldsymbol{W}_{\boldsymbol{g}_{src}} + \frac{\partial \mathcal{L}_{adv}^{low}}{\partial \boldsymbol{W}_{\boldsymbol{g}_{src}}}$

  Obtain $\mathcal{L}_{adv}$ according to Eq. (3)
  $\nabla \boldsymbol{W}_{\boldsymbol{g}_{tgt}} \leftarrow \nabla \boldsymbol{W}_{\boldsymbol{g}_{tgt}} + \frac{\partial \mathcal{L}_{adv}}{\partial \boldsymbol{W}_{\boldsymbol{g}_{tgt}}}$
  $\nabla \boldsymbol{W}_{\boldsymbol{g}_{share}} \leftarrow \nabla \boldsymbol{W}_{\boldsymbol{g}_{share}} + \frac{\partial \mathcal{L}_{adv}}{\partial \boldsymbol{W}_{\boldsymbol{g}_{share}}}$
  $\nabla \boldsymbol{W}_{\mathcal{C}} \leftarrow \nabla \boldsymbol{W}_{\mathcal{C}} + \frac{\partial \mathcal{L}_{adv}}{\partial \boldsymbol{W}_{\mathcal{C}}}$

  Obtain $\mathcal{L}_{sd}$ according to Eq. (13)
  $\nabla \boldsymbol{W}_{\boldsymbol{g}_{tgt}} \leftarrow \nabla \boldsymbol{W}_{\boldsymbol{g}_{tgt}} + \frac{\partial \mathcal{L}_{sd}}{\partial \boldsymbol{W}_{\boldsymbol{g}_{tgt}}}$
  $\nabla \boldsymbol{W}_{\boldsymbol{g}_{share}} \leftarrow \nabla \boldsymbol{W}_{\boldsymbol{g}_{share}} + \frac{\partial \mathcal{L}_{sd}}{\partial \boldsymbol{W}_{\boldsymbol{g}_{share}}}$
  $\nabla \boldsymbol{W}_{\mathcal{C}_{aux}} \leftarrow \nabla \boldsymbol{W}_{\mathcal{C}_{aux}} + \frac{\partial \mathcal{L}_{sd}}{\partial \boldsymbol{W}_{\mathcal{C}_{aux}}}$

  Obtain $\mathcal{L}_{d}^{low}$ according to Eq. (11)
  $\nabla \boldsymbol{W}_{\mathcal{D}_{low}} \leftarrow \nabla \boldsymbol{W}_{\mathcal{D}_{low}} + \frac{\partial \mathcal{L}_{d}^{low}}{\partial \boldsymbol{W}_{\mathcal{D}_{low}}}$

  Obtain $\mathcal{L}_{d}$ according to Eq. (12)
  $\nabla \boldsymbol{W}_{\mathcal{D}} \leftarrow \nabla \boldsymbol{W}_{\mathcal{D}} + \frac{\partial \mathcal{L}_{d}}{\partial \boldsymbol{W}_{\mathcal{D}}}$

  Update $\boldsymbol{W}_{\boldsymbol{g}_{src}}$, $\boldsymbol{W}_{\boldsymbol{g}_{tgt}}$, $\boldsymbol{W}_{\boldsymbol{g}_{share}}$, $\boldsymbol{W}_{\mathcal{C}}$, $\boldsymbol{W}_{\mathcal{C}_{aux}}$ with SGD
  Update $\boldsymbol{W}_{\mathcal{D}_{low}}$, $\boldsymbol{W}_{\mathcal{D}}$ with Adam

Table 1. Results on GTA5→Cityscapes with VGG16 and DeepLabv2. ST: self-training

Method	ST	*road*	*sw.*	*build*	*wall*	*fence*	*pole*	*light*	*sign*	*veg.*	*terrain*	*sky*	*person*	*rider*	*car*	*truck*	*bus*	*train*	*moto.*	*bicycle*	mIoU
SourceOnly		62.6	22.0	75.1	23.7	16.8	24.5	21.7	8.0	77.1	11.2	65.7	44.2	17.9	71.6	16.0	9.9	0.0	14.6	5.2	30.9
AdaptSeg [13]		87.3	29.8	78.6	21.1	18.2	22.5	21.5	11.0	79.7	29.6	71.3	46.8	6.5	80.1	23.0	26.9	0.0	10.6	0.3	35.0
AdaptSeg*(LS)*		86.8	31.9	78.7	27.8	17.6	20.5	22.3	8.0	79.0	23.3	69.8	46.9	16.6	80.1	23.3	18.6	3.0	13.7	5.9	35.3
CLAN [6]		88.0	30.6	79.2	23.4	20.5	**26.1**	23.0	14.8	**81.6**	**34.5**	**72.0**	45.8	7.9	80.5	**26.6**	**29.9**	0.0	10.7	0.0	36.6
AdvEnt [15]		86.9	28.7	78.7	28.5	**25.2**	17.1	20.3	10.9	80.0	26.4	70.2	47.1	8.4	**81.5**	26.0	17.2	**18.9**	11.7	1.6	36.1
FADA [16]		89.8	37.7	79.2	**30.5**	22.0	25.7	25.7	**15.1**	79.9	26.9	69.7	52.1	22.9	81.1	21.9	15.4	3.8	22.2	11.1	38.6
Ours		**90.1**	**43.0**	**79.5**	28.7	22.1	25.6	**27.4**	13.2	77.7	26.4	68.3	**52.8**	**25.3**	81.2	26.3	27.6	0.7	**23.7**	**20.4**	**40.0**
CBST [20]	✓	90.4	50.8	72.0	18.3	9.5	27.2	28.6	14.1	82.4	25.1	70.8	42.6	14.5	76.9	5.9	12.5	1.2	14.0	28.6	36.1
AdaptPatch [14]	✓	87.3	35.7	79.5	32.0	14.5	21.5	24.8	13.7	80.4	32.0	70.5	50.5	16.9	81.0	20.8	28.1	4.1	15.5	4.1	37.5
Label-Driven [17]	✓	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
FADA [16]	✓	92.3	51.1	83.7	33.1	**29.1**	28.5	28.0	21.0	82.6	32.6	**85.3**	55.2	**28.8**	83.5	24.4	37.4	0.0	21.1	15.2	43.8
Kim et al. [4]	✓	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
FDA-MBT [18]	✓	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
TPLD [11]	✓	83.5	49.9	72.3	17.6	10.7	29.6	28.3	9.0	78.2	20.1	25.7	47.4	13.3	79.6	3.3	19.3	1.3	14.3	33.5	34.1
DPL [2]	✓	89.2	44.0	83.5	**35.0**	24.7	27.8	38.3	25.3	**84.2**	39.5	81.6	54.7	25.8	83.3	29.3	**49.0**	5.2	30.2	32.6	46.5
Ours+ST	✓	**93.7**	**61.0**	**85.9**	31.1	28.4	**35.2**	**41.9**	24.8	82.0	27.3	83.7	**63.0**	27.2	**86.7**	**34.5**	48.2	**0.1**	**33.9**	**39.1**	**48.8**

Table 2. Results on Synthia→Cityscapes with VGG16 and DeepLabv2. ST: self-training. mIoU^+: mIoU of 13 classes

Method	ST	*road*	*sw.*	*build*	*wall*	*fence*	*pole*	*light*	*sign*	*veg.*	*sky*	*person*	*rider*	*car*	*bus*	*moto.*	*bicycle*	mIoU	mIoU^+
SourceOnly		7.2	14.7	48.7	**8.0**	0.2	17.0	1.3	4.3	71.6	77.7	46.1	5.9	66.3	18.4	1.9	3.9	24.6	28.3
AdaptSeg [13]		78.9	29.2	75.5	-	-	-	0.1	4.8	72.6	76.7	43.4	8.8	71.1	16.0	3.6	8.4	-	37.6
AdaptSeg*(LS)*		73.8	23.9	78.7	7.0	**0.7**	20.6	1.5	6.0	75.2	79.2	45.7	13.2	70.5	21.7	5.4	11.0	33.4	38.9
CLAN [6]		80.4	30.7	74.7	-	-	-	1.4	8.0	77.1	79.0	**46.5**	8.9	**73.8**	18.2	2.2	9.9	-	39.3
AdvEnt [15]		67.9	29.4	71.9	6.3	0.3	19.9	0.6	2.6	74.9	74.9	35.4	9.6	67.8	21.4	4.1	15.5	31.4	36.6
FADA [16]		**80.1**	**31.0**	**76.9**	5.4	0.5	**23.3**	**4.5**	7.8	**78.7**	78.2	41.9	12.6	68.3	21.7	6.9	15.6	34.6	40.3
Ours		78.7	27.7	76.1	**8.0**	**0.7**	20.5	1.5	**8.4**	74.7	**78.5**	43.6	**15.6**	70.3	**24.2**	**7.7**	**24.5**	**35.0**	**40.9**
CBST [20]	✓	69.6	28.7	69.5	12.1	0.1	25.4	11.9	13.6	82.0	81.9	49.1	14.5	66.0	6.6	3.7	32.4	35.4	36.1
AdaptPatch [14]	✓	72.6	29.5	77.2	3.5	0.4	21.0	1.4	7.9	73.3	79.0	45.7	14.5	69.4	19.6	7.4	16.5	33.7	39.6
Label-Driven [17]	✓	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	48.5
FADA [16]	✓	80.4	35.9	80.9	2.5	0.3	30.4	7.9	22.3	81.8	83.6	48.9	16.8	77.7	31.1	13.5	17.9	39.5	46.0
Kim et al. [4]	✓	**89.8**	**48.6**	78.9	-	-	-	0.0	4.7	80.6	81.7	36.2	13.0	74.4	22.5	6.5	32.8	-	43.8
FDA-MBT [18]	✓	84.2	35.1	78.0	6.1	0.4	27.0	8.5	22.1	77.2	79.6	55.5	19.9	74.8	24.9	14.3	40.7	40.5	47.3
TPLD [11]	✓	81.3	34.5	73.3	11.9	0.0	26.9	0.2	6.3	79.9	71.2	55.1	14.2	73.6	5.7	0.5	41.7	36.0	41.3
DPL [2]	✓	83.5	38.2	80.4	1.3	1.1	29.1	**20.2**	**32.7**	81.8	83.6	55.9	20.3	79.4	26.6	7.4	46.2	43.0	50.5
Ours+ST	✓	86.6	41.6	**82.6**	**17.4**	**5.0**	**31.7**	0.1	20.6	**83.2**	**87.1**	54.2	15.0	**82.9**	**40.4**	**27.0**	39.5	**44.7**	**50.8**

4 Class-wise Thresholds in SD

For Self-Discrimination, we use a class-wise threshold for each class to ignore the uncertain pixels in the pseudo labels. Specifically, the class-wise threshold is set to the p percentile of prediction confidences. However, the prediction confidences need to feed forward all target domain images, which is time-consuming and impractical for each training iteration. Instead, we do not calculate the prediction confidences of all target domain images. We first maintain a confidence queue (with a length limit of $100,000$ in our experiments) for each class, and then append prediction confidences of the current data batch into it at each training iteration. When the confidence queue is full, the earliest elements would be released to accommodate the new elements. In this way, we approximate the

Table 3. Results on Cityscapes→Cross-City with ResNet101 and DeepLabv2

City	Method	*road*	*sw.*	*build.*	*light*	*sign*	*veg.*	*sky*	*person*	*rider*	*car*	*bus*	*moto.*	*bicycle*	mIoU
Rome	SourceOnly	84.2	37.9	79.5	19.4	**49.8**	84.1	80.4	47.8	37.4	83.5	44.6	52.4	10.8	54.8
	Ours	86.3	**45.5**	83.7	19.4	46.9	86.4	92.6	58.4	47.6	**84.5**	39.8	**56.9**	**11.1**	58.4
	Ours+ST	**87.9**	41.9	**85.2**	**33.4**	41.4	**87.0**	**93.4**	**64.0**	**53.2**	83.9	**57.9**	51.1	9.2	**60.7**
Rio	SourceOnly	75.9	51.7	67.9	22.5	**31.7**	77.1	80.7	56.1	**39.6**	78.2	38.1	44.9	25.9	53.1
	Ours	80.0	59.3	**78.5**	12.5	27.9	**82.8**	87.5	63.6	36.0	**81.7**	36.0	**55.5**	31.5	56.4
	Ours+ST	**85.0**	**61.7**	77.0	**34.6**	26.4	82.6	**88.1**	**65.3**	36.9	81.4	**46.4**	46.9	**33.6**	**58.9**
Tokyo	SourceOnly	**84.9**	35.5	70.3	14.5	27.0	81.9	73.3	56.1	27.2	70.1	6.7	19.6	52.1	47.6
	Ours	84.8	**41.0**	74.9	28.1	31.1	83.0	**91.3**	61.0	32.1	72.2	**10.5**	**36.1**	59.7	54.3
	Ours+ST	84.5	37.5	**76.9**	**34.4**	**38.0**	**85.4**	**91.3**	**66.1**	**33.3**	**73.2**	7.6	29.7	**61.8**	**55.4**
Taipei	SourceOnly	84.9	35.6	76.6	27.2	**17.0**	74.9	82.8	30.0	31.6	70.4	33.9	54.7	22.7	49.4
	Ours	84.9	39.4	83.0	36.5	15.5	76.2	**94.8**	40.2	32.2	**72.7**	41.1	**58.3**	39.6	54.9
	Ours+ST	**86.0**	**40.2**	**86.5**	**51.1**	**17.0**	**77.6**	94.3	**45.7**	**33.2**	71.4	**54.3**	48.3	**40.2**	**57.4**

Table 4. Comparison with ProDA on first and second-round self-training

Method	GTA5→Cityscapes		Synthia→Cityscapes	
	First-round	Second-round	First-round	Second-round
ProDA [19]	53.6	54.9	52.3	53.9
Ours	**56.7**	**58.1**	**55.1**	**56.6**

prediction confidences of all target domain images by those in the confidence queue, and incur negligible computational cost in the meanwhile.

5 Extra Training Computational Cost

The computational cost comparison is demonstrated in Table 5. It shows that our method incurs negligible computational cost. Same environment is used for fair comparison. It is worth noting that our method does not introduce extra parameters during inference.

6 Plot Details

The plot in Fig. 1(c) of the submission file is conducted on GTA5→Cityscapes. Since we need to evaluate the performance on the source domain (mIoU_src) for plotting, we only employ the first $24,000$ samples of the GTA5 dataset for training, and the last 966 samples for validation. Note that for the normal experiments in Sec. 4 of the submission file, we still follow existing previous methods to use the total $24,966$ samples for training.

Similarly, we yield the plot on Synthia→Cityscapes in the same way, as shown in Fig. 1. From the plot, consistent conclusion can be made — our method can alleviate the issue of source domain overfitting.

Table 5. Computational cost comparison

Method	AdaptSegNet [13]	DecoupleNet+SD
Training Time	23h 22min	24h 45min
Training Parameters	54.72M	55.77M
Inference Parameters	51.94M	51.94M

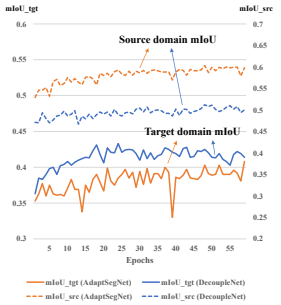

Fig. 1. Plot of the validation mIoU on source domain data (mIoU_src) by the dashed line and on target domain data (mIoU_tgt) by the solid line. Evaluation is made on Synthia→Cityscapes. This figure is similar to Fig. 1(c) of the submission file

7 Visual Comparison

The visual comparison of first-stage training and self-training are shown in Fig. 2 and Fig. 3, respectively. They demonstrate the superiority of our method. Compared to others, our approach tends to yield high-quality segmentation results with less artifacts.

Table 6. The results of adapting DecoupleNet to the UDA classification methods with ResNet50 on VisDA 2017 [7]

Method	Accuracy	Δ
MCD [10]	70.3	0.0
MCD + Ours	70.9	+0.6
DropToAdapt [5]	76.2	0.0
DropToAdapt + Ours	78.1	+1.9

8 Extension to UDA Classification Task

Our method can be easily adapted to the UDA classification task. To demonstrate the generalization ability, we adapt our method to two representative UDA

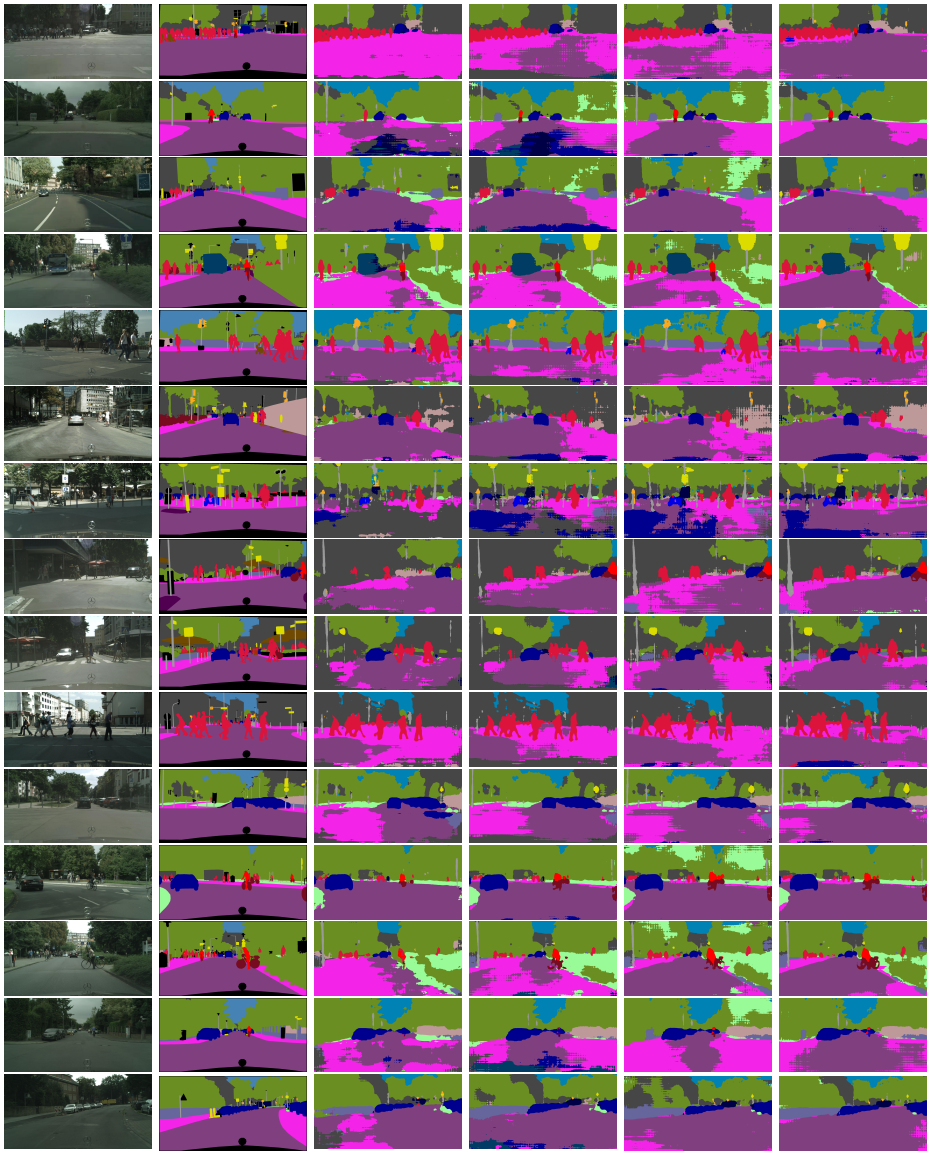

Fig. 2. Visual comparison with one-stage methods

classification methods, MCD [10] and DropToAdapt [5]. Similarly, we maintain two shallow blocks responsible for two domains respectively, while the deep blocks are shared by both domains. We also add the additional loss $\mathcal{L}_{adv}^{low}$ on the shallow source features ϕ_s. The pipeline and the losses used in MCD and

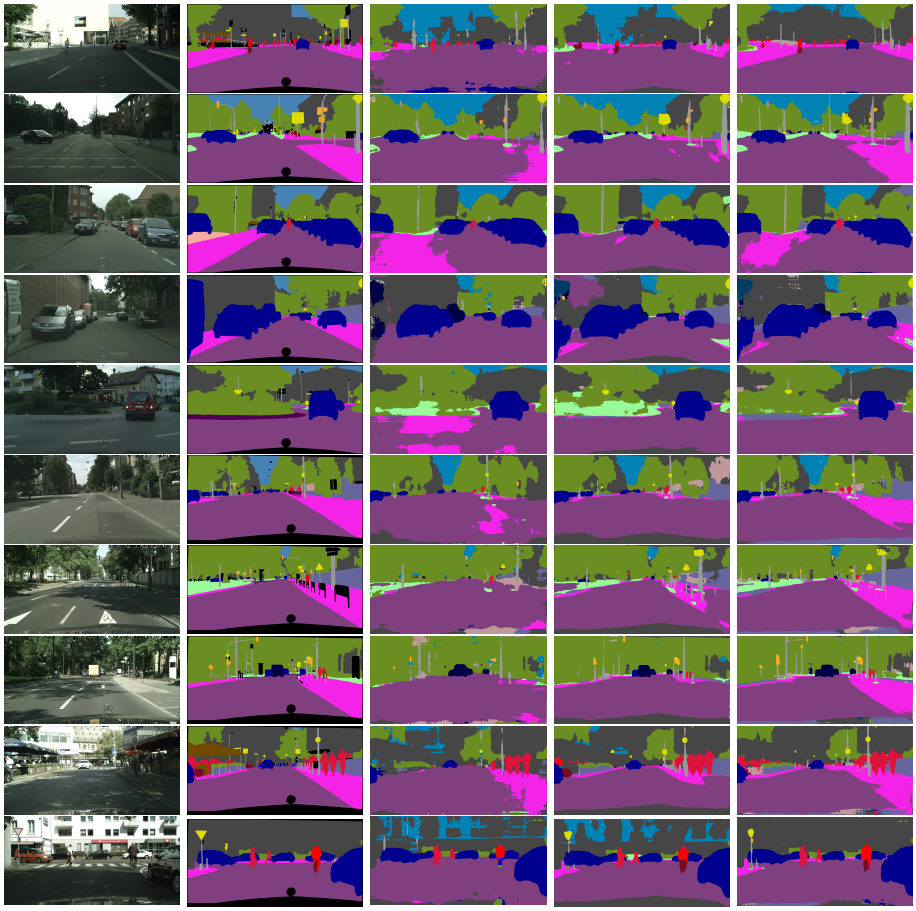

Fig. 3. Visual comparison with self-training based methods

DropToAdapt are all kept. Amazingly, as shown in Table 6, we can make further improvements on the UDA classification task.

9 More Implementation Details of Experiment

Experimental setting. During training, we follow [19] to perform random 512×892 crop. We use 'poly' learning rate decay policy where the base learning rate is scaled by $(1-iter/max\_iter)^{power}$ and $power$ is set to 0.9. We use SGD optimizer to train the segmentation model with weight decay and momentum set to 0.0005 and 0.9, respectively. The base learning rate is set to 0.00025 and 0.0025 for the backbone parameters and the others, respectively. To train discriminators, we use Adam optimizer with the base learning rate, beta1 and beta2 set to 0.0001, 0.9 and 0.99 respectively. Four NVIDIA GeForce RTX 2080Ti GPUs are used for training, and a training batch includes 8 source and 8 target domain images. The loss weights λ_{adv}, λ_{adv}^{low} and λ_{sd} are set to 0.01, 0.1 and 0.1 respectively. The ablation study is conducted on GTA5→Cityscapes with ResNet101 and DeepLabv2.

For the first stage, we train for $60,000$ iterations without any data augmentation except random cropping. As for the second stage, we train for $40,000$ iterations with the data augmentation of random color jitter. The class-wise threshold percentage is set to 0.8. To stabilize training, the model weights are initialized with the SourceOnly model.

Datasets. As shown in the submission file, our experiments are conducted on GTA5 → Cityscapes, Synthia → Cityscapes and Cityscapes → Cross-City. The details of these datasets are illustrated below.

The Cityscapes [3] dataset contains 2975 and 500 finely annotated urban scene images with 19 classes for training and validation respectively. Only the images in the training set are used during training. After training, the validation set is used to evaluate the performance of models.

The GTA5 [8] dataset is a synthetic dataset collected from a physically rendered video game called Grand Theft Auto V (GTAV). It comprises 24966 urban scene images, with compatible classes with Cityscapes. We only choose 19 semantic classes available in Cityscapes, while the rest pixels that belong to other classes are labeled as ignored.

The Synthia [9] dataset is also a synthetic urban scene dataset. We choose the SYNTHIA-RAND-CITYSCAPES subset, which comprises 9400 images but only shares 16 semantic classes with the Cityscapes dataset. Similar to the GTA5 dataset, the pixels that belong to other classes are also labeled as ignored.

The Cross-City [1] dataset is an urban scene dataset and contains 3,200 unlabeled images and 100 annotated images of Rome, Rio, Tokyo and Taipei. Its annotations share 13 classes with Cityscapes.

Network architecture. Since most segmentation networks can be generally divided into a feature encoder and a linear classifier, we modify the ASPP module

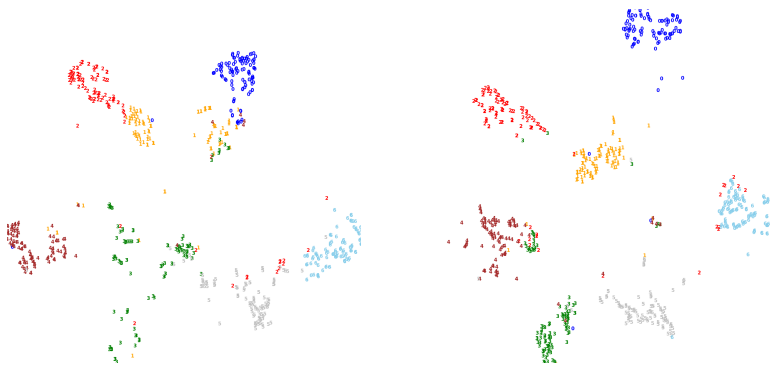

Fig. 4. The t-SNE visualizations of the target features w/o SD (Left) and w/ SD (Right). Each color represents each class. Note that both visualizations are in the same t-SNE space. More visualization details are described in the supplementary

in Deeplabv2 by first projecting multi-scale atrous features into a common dimension and then passing them into a linear classifier. This minor modification is for maintaining another auxiliary classifier in Self-Discrimination. For fair comparison, the modification incurs negligible parameters and does not affect the segmentation performance in our experiments. In Self-Discrimination, we maintain another linear classifier as the auxiliary classifier, so the additional training parameters brought by SD can be almost neglected.

10 Details of The t-SNE Visualizations

In Fig. 4, we demonstrate the t-SNE visualizations for the target features f_t of the models with and without SD. In particular, we firstly forward all the images in the Cityscapes validation set into the model that has already been trained with or without SD, obtaining the target feature f_t, whose shape is (H, W, C). Each feature vector in f_t has a corresponding ground-truth label, so we can obtain the feature vectors belonging to each class with the ground-truth labels. Further, we randomly select 100 feature vectors for each class, and then use the python sklearn package to perform t-SNE visualization for them, thus obtaining the Fig. 6 of the submission file. In this way, the figure is obtained from the whole validation set, which is more convincing because it alleviates the effect brought by random variance. Note that both results without (the Left) and with SD (the Right) are in the same t-SNE space. And to ensure the visibility, we choose 7 classes in the visualizations with each color representing each class, i.e., *road* (blue), *sidewalk* (orange), *building* (red), *person* (green), *car* (brown), *vegetation* (silver), *sky* (skyblue).

References

1. Chen, Y.H., Chen, W.Y., Chen, Y.T., Tsai, B.C., Frank Wang, Y.C., Sun, M.: No more discrimination: Cross city adaptation of road scene segmenters. In: ICCV (2017)
2. Cheng, Y., Wei, F., Bao, J., Chen, D., Wen, F., Zhang, W.: Dual path learning for domain adaptation of semantic segmentation. In: ICCV (2021)
3. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: CVPR (2016)
4. Kim, M., Byun, H.: Learning texture invariant representation for domain adaptation of semantic segmentation. In: CVPR (2020)
5. Lee, S., Kim, D., Kim, N., Jeong, S.G.: Drop to adapt: Learning discriminative features for unsupervised domain adaptation. In: ICCV (2019)
6. Luo, Y., Zheng, L., Guan, T., Yu, J., Yang, Y.: Taking a closer look at domain shift: Category-level adversaries for semantics consistent domain adaptation. In: CVPR (2019)
7. Peng, X., Usman, B., Kaushik, N., Hoffman, J., Wang, D., Saenko, K.: Visda: The visual domain adaptation challenge. arXiv:1710.06924 (2017)
8. Richter, S.R., Vineet, V., Roth, S., Koltun, V.: Playing for data: Ground truth from computer games. In: ECCV (2016)
9. Ros, G., Sellart, L., Materzynska, J., Vazquez, D., Lopez, A.M.: The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In: CVPR (2016)
10. Saito, K., Watanabe, K., Ushiku, Y., Harada, T.: Maximum classifier discrepancy for unsupervised domain adaptation. In: CVPR (2018)
11. Shin, I., Woo, S., Pan, F., Kweon, I.S.: Two-phase pseudo label densification for self-training based domain adaptation. In: ECCV (2020)
12. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556 (2014)
13. Tsai, Y.H., Hung, W.C., Schulter, S., Sohn, K., Yang, M.H., Chandraker, M.: Learning to adapt structured output space for semantic segmentation. In: CVPR (2018)
14. Tsai, Y.H., Sohn, K., Schulter, S., Chandraker, M.: Domain adaptation for structured output via discriminative patch representations. In: ICCV (2019)
15. Vu, T.H., Jain, H., Bucher, M., Cord, M., Pérez, P.: Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation. In: CVPR (2019)
16. Wang, H., Shen, T., Zhang, W., Duan, L.Y., Mei, T.: Classes matter: A fine-grained adversarial approach to cross-domain semantic segmentation. In: ECCV (2020)
17. Yang, J., An, W., Wang, S., Zhu, X., Yan, C., Huang, J.: Label-driven reconstruction for domain adaptation in semantic segmentation. In: ECCV (2020)
18. Yang, Y., Soatto, S.: Fda: Fourier domain adaptation for semantic segmentation. In: CVPR (2020)
19. Zhang, P., Zhang, B., Zhang, T., Chen, D., Wang, Y., Wen, F.: Prototypical pseudo label denoising and target structure learning for domain adaptive semantic segmentation. In: CVPR (2021)
20. Zou, Y., Yu, Z., Kumar, B., Wang, J.: Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In: ECCV (2018)