Supplementary Material for Maximum Classifier Discrepancy for Unsupervised Domain Adaptation

We would like to show supplementary information for our main paper. First, we introduce the detail of the experiments. Finally, we show some additional results of our method.

A. Toy Dataset Experiment

We show the detail of experiments on toy dataset in main paper.

A.1. Detail on experimental setting

The detail of experiment on toy dataset is shown in this section. When generating target samples, we set the rotation angle 30 in experiments of our main paper. We used Adam with learning rate 2.0×10^{-4} to optimizer the model. The batch size was set to 200. For a feature generator, we used 3-layered fully-connected networks with 15 neurons in hidden layer, in which ReLU is used as the activation function. For classifiers, we used three-layed fully-connected networks with 15 neurons in hidden layer and 2 neurons in output layer. The decision boundary shown in the main paper is obtained when we rotate the source samples 30 degrees to generate target samples. We set n to 3 in this experiment.

B. Experiment on Digit Dataset

We report the accuracy after training 20,000 iterations except for the adaptation between MNIST and USPS. Due to the lack of training samples of the datasets, we stopped training after 200 epochs (13 iterations per one epoch) to prevent over-fitting. We followed the protocol presented by [3] in the following three adaptation scenarios. **SVHN→MNIST** In this adaptation scenario, we used the standard training set as training samples, and testing set as testing samples both for source and target samples.

SYN DIGITS→SVHN We used 479400 source samples and 73257 target samples for training, 26032 samples for testing.

SYN SIGNS→GTSRB We randomly selected 31367 samples for target training and evaluated the accuracy on the rest.

MNIST↔USPS In this setting, we followed the different protocols provided by the paper, ADDA [9] and PixelDA [1]. The former protocol provides the setting where a part of training samples are utilized during training. 2,000 training samples are picked up for MNIST and 1,800 samples are used for USPS. The latter one allows to utilize all training samples during training. We utilized the architecture used as a classification network in PixelDA [1]. We added Batch Normalization layer to the architecture.

C. Experiment on VisDA Classification Dataset

The detail of architecture we used and the detail of other methods are shown in this section.

Class Balance Loss In addition to feature alignment loss, we used a class balance loss to improve the accuracy in this experiment. Please note that we incorporated this loss in comparable methods too. We aimed to assign the target samples to each classes equally. Without this loss, the target samples can be aligned in an unbalanced way. The loss is calculated as follows:

$$\mathbb{E}_{\mathbf{x_t} \sim X_t} \sum_{k=1}^{K} \log p(y = k | \mathbf{x}_t) \tag{1}$$

The constant term $\lambda = 0.01$ was multiplied to the loss and add this loss in Step 2 and Step 3 of our method. This loss was also introduced in MMD and DANN too when updating parameters of the networks.

For the fully-connected layers of classification networks, we set the number of neurons to 1000. In order to fairly compare our method with others, we used the exact the same architecture for other methods.

MMD We calculated the maximum mean discrepancy (MMD) [5], namely the last layer of feature generator networks. We used RBF kernels to calculate the loss. We used the the following standard deviation parameters:

$$\sigma = [0.1, 0.05, 0.01, 0.0001, 0.00001] \tag{2}$$

We changed the number of the kernels and their parameters, but we could not observe significant performance difference. We report the performance after 5 epochs. We could not see any improvement after the epoch.

DANN To train a model ([3]), we used two-layered domain classification networks. We set the number of neurons in the hidden layer as 100. We also used Batch Normalization, ReLU and dropout layer. Experimentally, we did not see any improvement when the network architecture is changed. According to the original method ([3]), learning rate is decreased every iteration. However, in our experiment, we could not see improvement, thus, we fixed learning rate 1.0×10^{-3}. In addition, we did not introduce gradient reversal layer for our model. We separately update discriminator and generator. We report the accuracy after 1 epoch.

D. Experiments on Semantic Segmentation

We describe the details of our experiments on semantic segmentation.

D.1. Details

Datasets GTA [7], Synthia [8] and Cityscapes [2] are vehicle-egocentric image datasets but GTA and Synthia are synthetic and Cityscapes is real world dataset. GTA is collected from the open world in the realistically rendered computer game Grand Theft Auto V (GTA, or GTA5). It contains 24,996 images, whose semantic segmentation annotations are fully compatible with the classes used in Cityscapes. Cityscapes is collected in 50 cities in Germany and nearby countries. We only used dense pixel-level annotated dataset collected in 27 cities. It contains 2,975 training set, 500 validation set, and 1525 test set. We used training and validation set. Please note that the labels of Cityscapes are just used for evaluation and never used in training. Similarly, we used the training splits of Synthia dataset to train our model.

Training Details When training, we ignored the pixel-wise loss that is annotated *backward* (*void*). Therefore, when testing, no predicted *backward* label existed. The weight decay ratio was set to 2×10^{-5} and we used no augmentation methods.

Network Architecture We applied our method to FCN-8s based on VGG-16 network. Convolution layers in original VGG-16 networks are used as generator and fully-connected layers are used as classifiers. For DRN-D-105, we followed the implementation of https://github.com/fyu/drn. We applied our method to dilated residual networks [10, 11] for base networks. We used *DRN-D-105* model. We used the last convolution networks as classifier networks. All of lower layers are used as a generator.

Evaluation Metrics As evaluation metrics, we use intersection-over-union (IoU) and pixel accuracy. We use the evaluation code[1] released along with VisDA challenge [6]. It calculates the PASCAL VOC intersection-over-union, i.e., IoU $= \frac{\text{TP}}{\text{TP+FP+FN}}$, where TP, FP, and FN are the numbers of true positive, false positive, and false negative pixels, respectively, determined over the whole test set. For further discussing our result, we also compute pixel accuracy, pixelAcc. $= \frac{\sum_i n_{ii}}{\sum_i t_i}$, where n_{ii} denotes number of pixels of class i predicted to belong to class j and t_i denotes total number of pixels of class i in ground truth segmentation.

[1] https://github.com/VisionLearningGroup/taskcv-2017-public/blob/master/segmentation/eval.py

E. Additional Results

E.1. Training via Gradient Reversal Layer

In our main paper, we provide the training procedure that consists of three training steps and the number of updating generator (k) is a hyper-parameter in our method. We found that introducing gradient reversal layer (GRL) [3] enables to update our model in only one step and works well in many settings. This improvement makes training faster and deletes hyper-parameter in our method. We provide the detail of the improvement and some experimental results here.

Training Procedure We simply applied gradient reversal layer when updating classifiers and generator in an adversarial manner. The layer flips the sign of gradients when back-propagating the gradient. Therefore, update for maximizing the discrepancy via classifier and minimizing it via generator was conducted simultaneously. We publicize the code with this implementation.

Results The experimental results on semantic segmentation are shown in Table 1,2, and Fig. 1. Our model with GRL shows the same level of performance compared to the model trained with our proposed training procedure.

E.2. Sensitivity to Hyper-Parameter

The number of updating generator is the hyper-parameter peculiar to our method. Therefore, we show additional experimental results related to it. We employed the adaptation from SVHN to MNIST and conducted experiments where $n = 5, 6$. The accuracy was 96.0% and 96.2% on average. The accuracy seems to increase as we increase the value though it saturates. Training time required to obtain high accuracy can increase too. However, considering the results of GRL on semantic segmentation, the relationship between the accuracy and the number of n seems to depend on which datasets to adapt.

References

[1] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *CVPR*, 2017. 1

[2] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The

Network	method	mIoU	road	sdwk	bldng	wall	fence	pole	light	sign	vgttn	trrn	sky	person	rider	car	truck	bus	train	mcycl	bcycl
VGG-16	Source Only	24.9	25.9	10.9	50.5	3.3	12.2	25.4	28.6	13.0	78.3	7.3	63.9	52.1	7.9	66.3	5.2	7.8	0.9	13.7	0.7
	FCN Wld [4]	27.1	70.4	32.4	62.1	14.9	5.4	10.9	14.2	2.7	79.2	21.3	64.6	44.1	4.2	70.4	8.0	7.3	0.0	3.5	0.0
	CDA (I) [12]	23.1	26.4	10.8	69.7	10.2	9.4	20.2	13.6	14.0	56.9	2.8	63.8	31.8	10.6	60.5	10.9	3.4	**10.9**	3.8	9.5
	Ours (k=2)	28.0	87.4	15.4	75.5	17.4	9.9	16.2	11.9	0.6	80.6	28.1	60.2	32.5	0.9	75.4	13.6	4.8	0.1	0.7	0.0
	Ours (k=3)	27.3	86.0	10.5	75.1	20.0	2.9	19.4	8.4	0.7	78.4	19.4	74.8	23.2	0.3	74.1	14.3	10.4	0.2	0.1	0.0
	Ours (k=4)	28.8	86.4	8.5	76.1	18.6	9.7	14.9	7.8	0.6	82.8	32.7	71.4	25.2	1.1	76.3	16.1	17.1	1.4	0.2	0.0
	Ours (GRL)	27.3	86.2	16.1	74.4	20.7	9.5	21.5	14.8	0.1	80.4	27.8	50.3	33.9	1.2	67.6	10.8	3.0	0.2	0.9	0.0
DRN-105	Source Only	22.2	36.4	14.2	67.4	16.4	12.0	20.1	8.7	0.7	69.8	13.3	56.9	37.0	0.4	53.6	10.6	3.2	0.2	0.9	0.0
	DANN [3]	32.8	64.3	23.2	73.4	11.3	18.6	29.0	31.8	14.9	82.0	16.8	73.2	53.9	12.4	53.3	20.4	11.0	5.0	18.7	9.8
	Ours (k=2)	39.7	90.3	31.0	78.5	19.7	17.3	28.6	30.9	16.1	83.7	30.0	69.1	**58.5**	**19.6**	81.5	23.8	**30.0**	5.7	**25.7**	**14.3**
	Ours (k=3)	38.9	**90.8**	**35.6**	**80.5**	22.9	15.5	27.5	24.9	15.1	84.2	31.8	77.4	54.6	17.2	82.0	21.6	29.0	1.3	21.8	5.3
	Ours (k=4)	38.1	89.2	23.2	80.2	**23.6**	18.1	27.7	25.0	9.3	**84.4**	**34.6**	**79.5**	53.2	16.0	**84.1**	**26.0**	22.5	5.2	16.7	4.8
	Ours (GRL)	**39.9**	90.4	34.5	79.3	20.4	**20.9**	**33.1**	28.3	**18.5**	82.4	22.6	75.5	57.6	18.6	82.7	24.1	25.6	7.6	23.9	12.3

Table 1. Adaptation results on the semantic segmentation. We evaluate adaptation from GTA5 to Cityscapes dataset.

Network	method	mIoU	road	sdwlk	bldng	wall	fence	pole	light	sign	vgttn	sky	prsn	ridr	car	bus	mcycl	bcycl
VGG-16	Source Only [12]	22.0	5.6	11.2	59.6	0.8	0.5	21.5	8.0	5.3	72.4	75.6	35.1	9.0	23.6	4.5	0.5	18.0
	FCN Wld [4]	20.2	11.5	19.6	30.8	4.4	0.0	20.3	0.1	11.7	42.3	68.7	51.2	3.8	54.0	3.2	0.2	0.6
	CDA (I+SP) [12]	29.0	65.2	26.1	74.9	0.1	0.5	10.7	3.7	3.0	76.1	70.6	47.1	8.2	43.2	20.7	0.7	13.1
DRN_105	Source Only	23.4	14.9	11.4	58.7	1.9	0.0	24.1	1.2	6.0	68.8	76.0	54.3	7.1	34.2	15.0	0.8	0.0
	DANN [3]	32.5	67.0	29.1	71.5	14.3	0.1	28.1	12.6	10.3	72.7	76.7	48.3	12.7	62.5	11.3	2.7	0.0
	Ours (k=2)	36.3	83.5	40.9	77.6	6.0	0.1	27.9	6.2	6.0	83.1	83.5	51.5	11.8	78.9	19.8	4.6	0.0
	Ours (k=3)	37.3	84.8	43.6	79.0	3.9	0.2	29.1	7.2	5.5	83.8	83.1	51.0	11.7	79.9	27.2	6.2	0.0
	Ours (k=4)	37.2	88.1	43.2	79.1	2.4	0.1	27.3	7.4	4.9	83.4	81.1	51.3	10.9	82.1	29.0	5.7	0.0
	Ours (GRL)	34.8	74.7	35.5	75.9	6.2	0.1	29.0	7.4	6.1	82.9	83.4	47.8	9.2	71.7	19.3	7.0	0.0

Table 2. Adaptation results on the semantic segmentation. We evaluate adaptation from Synthia to Cityscapes dataset.

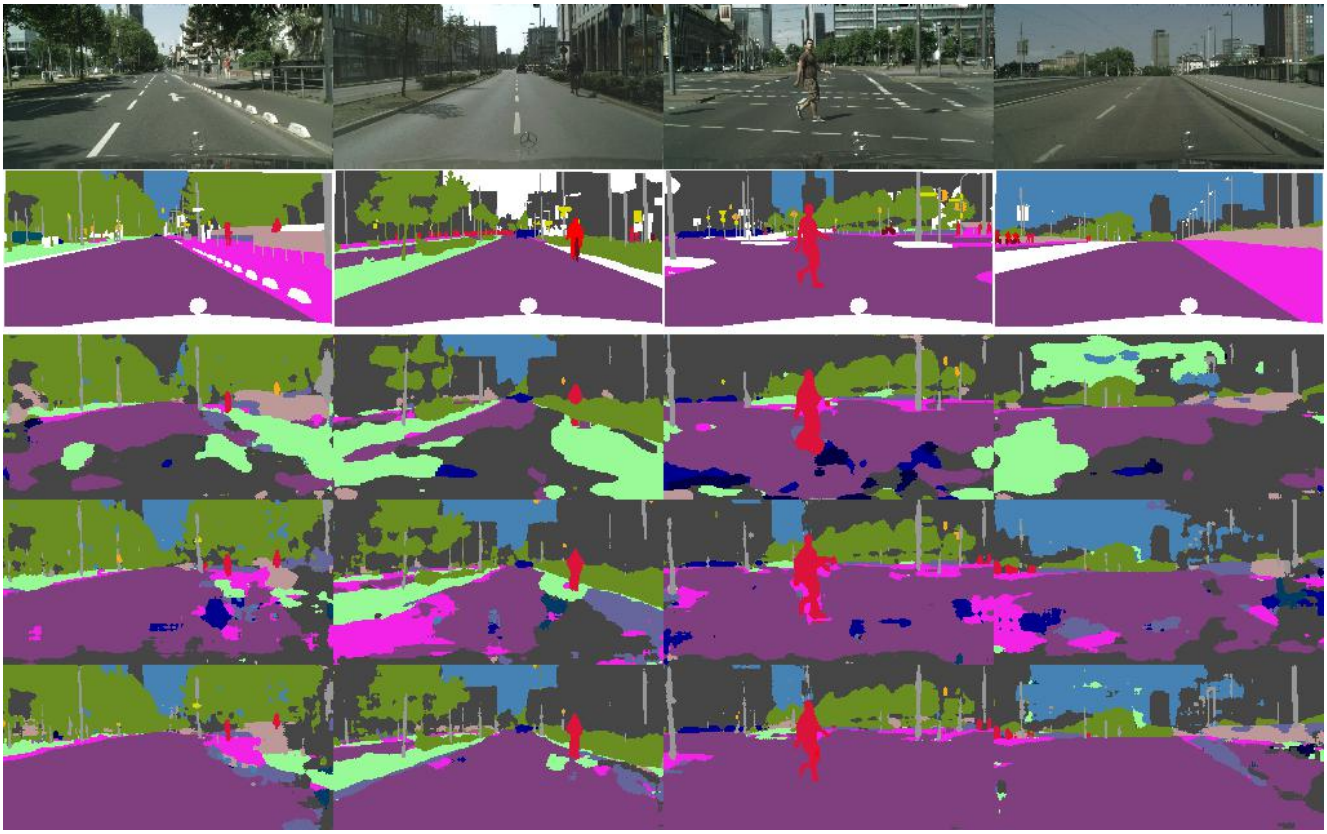

Figure 1. Qualitative results on adaptation from GTA5 to Cityscapes. From top to bottom, input, ground truth, result of source only model, DANN, and our proposed method.

cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 2

[3] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, 2014. 1, 2, 3

[4] J. Hoffman, D. Wang, F. Yu, and T. Darrell. Fcns in the wild: Pixel-level adversarial and constraint-based adapta-

tion. *arXiv:1612.02649*, 2016. 3

[5] M. Long, Y. Cao, J. Wang, and M. I. Jordan. Learning transferable features with deep adaptation networks. In *ICML*, 2015. 1

[6] X. Peng, B. Usman, N. Kaushik, J. Hoffman, D. Wang, and K. Saenko. Visda: The visual domain adaptation challenge. *arXiv:1710.06924*, 2017. 2

[7] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for data: Ground truth from computer games. In *ECCV*, 2016. 2

[8] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *CVPR*, 2016. 2

[9] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation. In *CVPR*, 2017. 1

[10] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016. 2

[11] F. Yu, V. Koltun, and T. Funkhouser. Dilated residual networks. In *CVPR*, 2017. 2

[12] Y. Zhang, P. David, and B. Gong. Curriculum domain adaptation for semantic segmentation of urban scenes. In *ICCV*, 2017. 3