Supplemental Material
Strong-Weak Distribution Alignment for Adaptive Object Detection

A. Network Architecture

We used the same architecture of the domain classifier for Faster RCNN with ResNet101 and VGG16. As the number of the channels in input features is different, we changed the channel size of network according to the backbone network.

In case of VGG16 model, the feature in $conv3\_3$ layer is fed into the local domain classifier. The feature in the last $res2c$ layer is fed into the local domain classifier with regard to ResNet101 model. The name of the layer is cited by the Caffe [1] prototxt.

Table 1. The architecture of the domain classifiers.

Global Domain Classifier
Conv $3 \times 3 \times 512$, stride 2, pad 1
Batch Normalization, ReLU, Dropout
Conv $3 \times 3 \times 128$, stride 2, pad 1
Batch Normalization, ReLU, Dropout
Conv $3 \times 3 \times 128$, stride 2, pad 1
Batch Normalization, ReLU, Dropout
Average Pooling
Fully connected 128×2
Softmax

Local Domain Classifier
Conv $1 \times 1 \times 256$, stride 1, pad 0
ReLU
Conv $1 \times 1 \times 128$, stride 1, pad 0
ReLU
Conv $1 \times 1 \times 1$, stride 1, pad 0
Sigmoid

Global Domain Classifier

The global domain classifier has three layered convolution layers, global average pooling and one Linear layer. The kernel size of the convolution layers is set as three. Batch Normalization, ReLU, and dropout layers are attached after each convolution layer. The output of the global domain classifier is activated by softmax function. Context vector is extracted after the average pooling layer. Therefore, the vector has 128 dimensions.

Local Domain Classifier

The local domain classifier has three layered convolution layers. The kernel size of the convolution layers is set as one. The output of the local domain classifier is activated by sigmoid function. Context vector is extracted before the last convolution layer. Therefore, the vector has 128 dimensions.

B. Pixel-level Adaptation

The results on a model trained with images generated by CycleGAN were provided in our paper. We describe the details of how we trained the model.

Training of CycleGAN

To train the CycleGAN, we used the Pytorch implementation https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix. We used all training images in both domains. We trained CycleGAN for 10 epochs and employed the source images translated into the target domain for training our Faster RCNN model.

Training of Faster RCNN

We found that some of the translated images are not translated correctly. Cars are completely hidden by large noise. In order to suppress the effect of such corrupted images, we trained our model using source, translated source and target images. Translated images are utilized just for training detection modules and not utilized for domain classification. Namely, we have three images in each mini-batch, source, translated source and target image. Source and target images are used as we mentioned in our main paper. The translated source one is used to calculate and back-propagate the detection loss.

C. Additional Results

Results on source domain

Table 2, 5, and 6 show the results on source domain in three adaptation scenarios. In all scenarios, our method does not significantly degrade the detection performance on the source domain.

Results on target domain

Table 3, 5 and 7 provide results including local-level only adaptation and pixel-level adaptation. We did not show the pixel-level adaptation results on Clipart and Watercolor dataset in our main paper. With regard to Cityscape, we show more ablations in this table. In adaptation for clipart dataset, training with the images generated by CycleGAN does not improve the performance. The possible reasons are that the style of the target images is largely different and that target images have diverse styles of examples. Then, CycleGAN may not generate images suitable for adaptive detection. In the experiments on Watercolor, the performance greatly improved with the use of pixel-level adaptation. The model demonstrates almost oracle-level performance. Table 4 denotes the results when using VGG network for the adaptation from PASCAL VOC to clipart. The performance largely improved with our method.

Parameter Sensitivity

Fig. 1(a) presents the sensitivity to the parameter λ in Eq. 12 in our main paper. The parameter is the trade-off between training of detection and adversarial training. Our method performed better than the baseline domain classifier in all values ranging from 0 to 1.

Fig. 1(b) presents the sensitivity to the parameter γ of Focal Loss. The result is obtained in adaptation from Sim10k to Cityscape. The parameter controls how strictly we align features between domains. As the parameter gets small, the domain classifier will look at all examples. As shown in the figure, the peak of the performance was around 3.0, in which AP was 42.3.

Domain Evidence

Fig. 2 is the additional domain evidence visualization. Although the behavior differs from dataset to dataset, the feature extractor tries to partially fool the domain classifier.

Table 2. Results on PASCAL VOC in adaptation from PASCAL VOC to Clipart Dataset. Average precision (%) is evaluated on PASCAL. Our method does not degrade the performance on the source whereas BDC-Faster and DC-Faster degrade it.

Method	G	I	C	L	aero	bcycle	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	hrs	bike	prsn	plnt	sheep	sofa	train	tv	MAP
Faster RCNN					77.7	80.3	82.5	**79.0**	**68.0**	88.1	85.7	87.0	53.1	**87.3**	58.2	88.3	85.0	87.9	**80.5**	**52.8**	75.9	69.4	86.3	77.8	77.5
BDC-Faster	✓				77.6	80.0	77.6	61.8	61.3	83.0	**86.1**	86.5	50.9	79.8	59.8	84.7	82.2	79.5	78.1	45.3	73.4	70.1	80.8	73.8	73.6
DA-Faster	✓	✓			64.5	70.6	66.4	62.9	49.9	77.2	78.0	70.7	44.7	78.3	56.1	70.8	75.6	85.2	74.5	37.2	65.4	60.2	72.3	66.9	66.4
Proposed	✓				**79.5**	80.7	82.5	78.6	62.2	86.1	85.4	87.5	**60.2**	79.0	**68.8**	88.6	86.2	**88.7**	78.9	52.5	78.5	**71.6**	**87.9**	76.9	**78.0**
	✓		✓		78.6	81.7	**83.4**	74.7	62.9	86.9	85.4	**90.6**	56.8	85.7	57.9	89.0	87.4	87.7	78.1	52.3	**79.3**	67.6	87.6	**79.5**	77.6
	✓		✓	✓	77.5	**84.7**	81.1	71.1	63.8	**88.5**	84.7	87.7	54.5	81.6	60.8	**89.4**	**87.8**	88.2	78.2	49.9	78.9	70.2	82.0	**79.5**	77.0

Table 3. Results on adaptation from PASCAL VOC to Clipart Dataset. Average precision (%) is evaluated on target images. G, I, CTX, L, P indicate global alignment, instance-level alignment, context-vector based regularization, local-alignment and pixel-alignment respectively. Faster RCNN* indicates Faster RCNN trained on source images and source images translated by CycleGAN.

Method	G	I	CTX	L	P	aero	bcycle	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	hrs	bike	prsn	plnt	sheep	sofa	train	tv	MAP
Faster RCNN						**35.6**	52.5	24.3	23.0	20.0	43.9	32.8	10.7	30.6	11.7	13.8	6.0	**36.8**	45.9	48.7	41.9	**16.5**	7.3	22.9	32.0	27.8
Faster RCNN*					✓	26.4	52.7	28.3	24.1	28.5	49.7	30.2	13	35.3	26.5	15.8	7.6	26.1	68.1	47.2	42.5	5.9	23.2	41	42.6	31.7
BDC-Faster	✓					20.2	46.4	20.4	19.3	18.7	41.3	26.5	6.4	33.2	11.7	**26.0**	1.7	36.6	41.5	37.7	44.5	10.6	20.4	33.3	15.5	25.6
DA-Faster	✓	✓				15.0	34.6	12.4	11.9	19.8	21.1	23.2	3.1	22.1	26.3	10.6	10.0	19.6	39.4	34.6	29.3	1.0	17.1	19.7	24.8	19.8
Proposed	✓					30.5	48.5	**33.6**	24.8	41.2	48.9	32.4	17.2	34.5	55.0	19.0	13.6	35.1	66.2	**63.0**	45.3	12.5	22.6	45.0	38.9	36.4
				✓		19.8	50.7	25.4	21.7	30.2	47.2	27.1	8.5	33.5	26.8	14.0	11.7	31.5	62.0	49.9	39.6	9.1	23.8	39.5	38.4	30.5
	✓		✓			31.7	**55.2**	30.9	26.8	**43.4**	47.5	**40.0**	7.9	**36.7**	50.0	14.3	**18.0**	29.2	**68.1**	62.3	50.4	13.4	24.5	**54.2**	45.8	37.5
	✓		✓	✓		26.2	48.5	32.6	**33.7**	38.5	**54.3**	37.1	**18.6**	34.8	**58.3**	17.0	12.5	33.8	65.5	61.6	**52.0**	9.3	**24.9**	54.1	**49.1**	**38.1**
	✓		✓	✓	✓	31.1	53.7	28.9	24.9	40.3	49.0	38.1	14.6	41.9	43.8	15.3	7.2	27.9	75.5	57.3	41.8	6.7	23.3	48.5	44.1	35.7

Table 4. Results on adaptation from PASCAL VOC to Clipart Dataset with VGG. Average precision (%) is evaluated on target images.

Method	G	CTX	L	aero	bcycle	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	hrs	bike	prsn	plnt	sheep	sofa	train	tv	MAP
Faster RCNN				15.7	31.9	22.4	8.2	38.8	59.4	17.8	6.6	37.0	5.7	12.7	7.2	17.4	49.0	36.0	32.1	11.2	2.9	29.8	28.4	23.5
BDC-Faster	✓			11.0	40.9	12.2	10.1	28.9	29.2	28.0	6.0	23.5	8.8	13.1	6.5	22.4	45.6	46.9	35.9	9.7	9.3	18.9	20.9	21.4
Proposed	✓			18.6	45.0	22.2	23.2	23.9	21.1	28.6	5.2	31.8	39.1	19.7	0.9	25.2	56.1	54.3	36.1	27.8	6.8	32.4	40.4	27.9
	✓	✓	✓	16.0	53.2	27.5	21.6	32.0	48.4	32.4	12.2	32.5	27.3	12.3	13.1	24.3	62.4	55.5	41.2	21.0	13.2	37.8	46.1	31.5

Table 5. Results on adaptation from PASCAL VOC to WaterColor Dataset (%). Left: AP evaluated on PASCAL VOC. Right: AP evaluated on WaterColor. AP on adaptation from PASCAL VOC to WaterColor (%). The definition of G, I, CTX, L is the same as defined in the main paper. Faster RCNN* indicates Faster RCNN trained by source images and translated source images by CycleGAN.

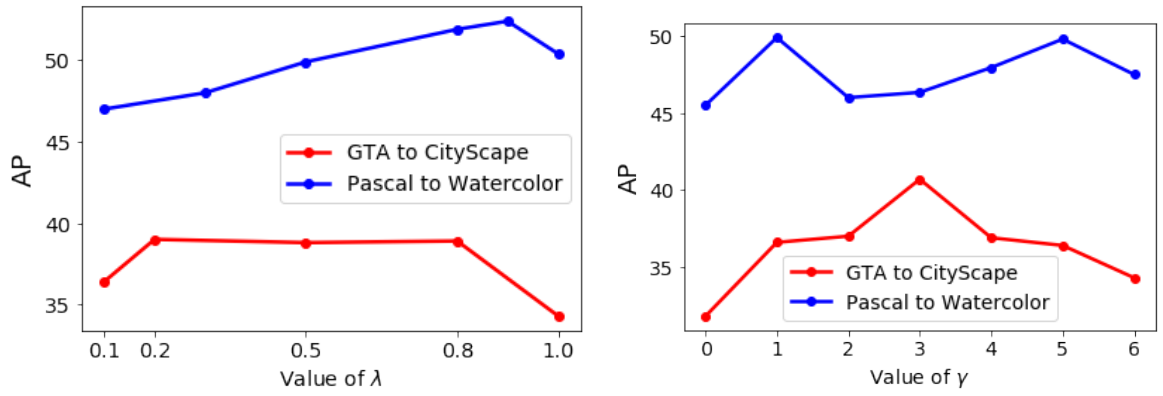
						AP on a source domain							AP on a target domain						
Method	G	I	CTX	L	P	bike	bird	car	cat	dog	prsn	MAP	bike	bird	car	cat	dog	prsn	MAP
Faster RCNN						82.1	82.3	86.5	89.3	85.6	84.3	85.0	68.8	46.8	37.2	32.7	21.3	60.7	44.6
Faster RCNN*					✓	79.7	82.7	86	88.9	85	82.1	84.1	83.3	52.7	45.3	33.1	28.8	64	51.2
BDC-Faster	✓					80.9	82.6	86.4	87.9	82.7	83.3	84.0	68.6	48.3	47.2	26.5	21.7	60.5	45.5
DA-Faster	✓	✓				75.7	84.4	84.5	88.5	83.6	81.6	83.1	75.2	40.6	48.0	31.5	20.6	60.0	46.0
	✓					79.4	84.6	85.8	89.6	85.7	84.1	84.9	66.4	53.7	43.8	37.9	31.9	65.3	49.8
				✓		80.0	82.0	84.7	86.8	83.6	81.3	83.1	79.4	54.8	47.2	37.1	31.5	62.4	52.1
Proposed	✓		✓			79.5	84.6	86.1	89.2	84.5	82.8	84.4	71.3	52.0	46.6	36.2	29.2	67.3	50.4
	✓		✓	✓		79.8	87.4	85.5	88.1	84.5	84.0	84.9	82.3	**55.9**	46.5	32.7	35.5	66.7	53.3
	✓		✓	✓	✓	78.9	83.1	84.2	87.4	85.6	82.8	83.7	**90.5**	54.8	**49.4**	**38.6**	**38.8**	**67.9**	**56.7**
Oracle						82.1	82.3	86.5	89.3	85.6	84.3	85.0	83.6	59.4	50.7	43.7	39.5	74.5	58.6

Table 6. Results on adaptation from Cityscape to FoggyCityscape Dataset (%). The performance is evaluated on Cityscape.

					AP on a source domain								
Method	G	I	CTX	L	bus	bcycl	car	mcycl	prsn	rider	train	truck	MAP
Faster RCNN					55.6	39.0	52.3	38.8	33.7	47.7	39.1	33.1	42.4
NDC-Faster	✓				56.4	38.2	52.7	36.5	33.6	49.3	41.9	32.0	42.6
DA-Faster	✓	✓			58.3	38.8	52.6	42.5	35.1	47.6	44.1	34.2	44.2
	✓				62.6	37.9	52.2	35.1	35.0	48.5	47.7	34.9	44.2
Proposed	✓		✓		57.0	39.3	52.3	39.9	33.6	48.3	41.6	36.0	43.5
	✓		✓	✓	57.9	39.4	52.6	39.4	35.2	48.3	47.7	37.4	44.7

Table 7. Results on adaptation from Sim10k to Cityscape Dataset (%). Average precision is evaluated on target images. Faster RCNN* indicates Faster RCNN trained by source images and translated source images by CycleGAN.

Method	G	I	CTX	L	P	AP on Car
Faster RCNN						34.6
Faster RCNN*					✓	40.0
BDC-Faster	✓					31.8
DA-Faster	✓	✓				34.2
	✓					36.4
				✓		40.2
Proposed (FL)	✓		✓			38.2
	✓		✓	✓		40.1
	✓		✓		✓	41.5
	✓		✓	✓	✓	40.7
Proposed Method with different parameters						
FL ($\gamma = 3$)	✓		✓			42.3
Oracle						53.1



(a) Parameter sensitivity to λ (b) Parameter sensitivity to γ

Figure 1. Parameter sensitivity to the value of λ (Left) and γ (Right) in adaptation from Sim10k to Cityscape and from Pascal to Watercolor.

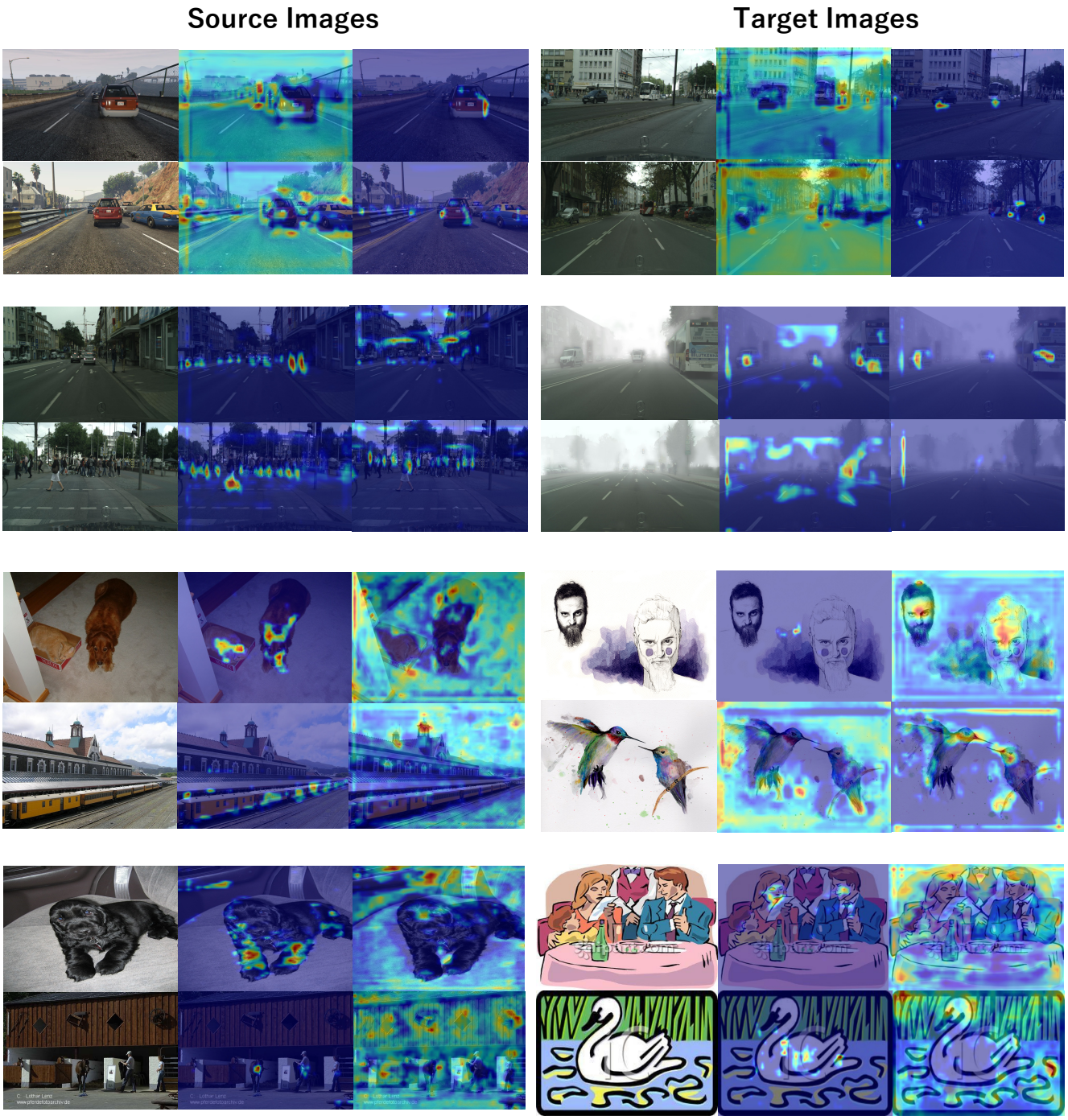


Figure 2. Visualization of domain evidence using Grad-Cam. The evidence is obtained by the global-domain classifier. The pictures show results on target and source images respectively. From left to right, input images, images of evidence for the target, evidence of the source domain. The behavior of the domain classifier seems to be different in the adaptation scenarios. However, the feature extractor tries to partially fool the domain classifier.

References

[1] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACMMM*, 2014. 1