6 Improving Object Detection via Local-global Contrastive Learning: Supplementary materials

We provide additional materials to supplement our main paper. In Sec. 6.1 we report additional qualitative results to complement those found in the main paper. Sec. 6.2 provides a sensitivity analysis and further extended ablative study on method components for various detector backbone architectures. In Sec. 6.3 we provide a comparison with contrastive learning based I2I translation methods. Finally, Sec. 6.4 gives supplementary information on learning hyperparameters and further implementation details.

6.1 Additional qualitative results

We show additional visualisations of the learned attention masks, for our two instantiations of G_A, in Fig. S1. We observe that in the supervised case the attention masks learn to explicitly focus on detection target instances. In the self-supervised local-global case, even if foreground / background separation in relation to detection targets is less clear, we find the model is still able to learn to disentangle the semantic content and focus on regions that contain objects. We illustrate further qualitative detection performance results, under the three adaptation scenarios studied in the main paper, in Figs. S3, S4 and S2. The provided examples further illustrate adaptation gains and the ability of the proposed method to improve cross domain detection performance.

6.2 Sensitivity analysis

We examine the impact of the proposed components in detail and report results in Tab. S1. We select the Foggy Cityscapes $\rightarrow$ Cityscapes adaptation scenario and train the proposed model architecture under the following ablations; (i) *without* the G_A network and *without* the proposed attention module, (ii) *with* the G_A network, *with* the proposed attention module and *without* loss $\mathcal{L}_{G_A}$, (iii) *with* the G_A network, *with* the proposed attention module and *with* an unsupervised $\mathcal{L}_{G_A}$ loss and finally (iv) *with* the G_A network, *with* the proposed attention module and *with* a supervised $\mathcal{L}_{G_A}$ loss. All models are trained under identical settings which are reported in Sec 6.4. In all cases we use the adversarial loss $\mathcal{L}_{adv}$ found in Eq. (3) and the InfoNCE loss found in Eq. (1) of the main paper.

We observe that detection performance is lower in case (i) where all method components under consideration are absent. In case (ii) performance is improved by 0.5–1.7% mAP@.5 which we attribute to the addition of the proposed attention module, trained without any guidance (i.e. $\mathcal{L}_{G_A} = 0$). Inclusion of all components results in 2.3–2.6% mAP@.5 gains for the unsupervised model (case (iii)) and 2.3–4% mAP@.5 gains for the supervised model (case (iv)). We note that when using the Res-Net-101-FPN backbone, our unsupervised model (mAP@.5 49.1) outperforms its supervised counterpart (mAP@.5 48.0), highlighting the potential of our unsupervised proposal. Our unsupervised loss not only guides the attention generator towards disentangling background and foreground, but additionally improves representation learning at the level of the encoder network. We conjecture that this is more effective on higher capacity networks such as Res-Net-101. Finally, the ablation study provides quantitative evidence towards verifying the efficacy of the individual components under the proposed method.

We conduct a further sensitivity study on detector training settings and backbone model

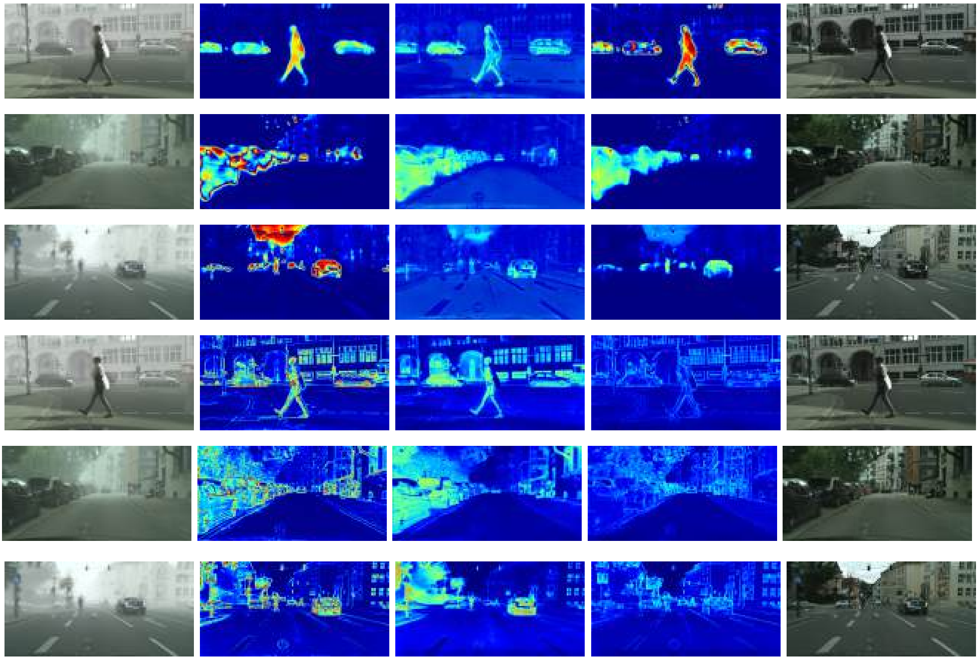

Figure S1: Visualization of the learned foreground attention masks of the proposed supervised model (rows 1-4) and self-supervised local-global model (rows 5-8). Column 1 shows the input (foggy weather) image, columns 2–4 visualize attention masks from 3 different channels A_k and column 5 shows the translated (clean weather) result.

Figure S2: The Foggy Cityscapes $\rightarrow$ Cityscapes adaptation scenario. In all cases the detector model is trained on the Cityscapes dataset and evaluated on Foggy Cityscapes. We visualise detection inference results on Foggy Cityscapes imagery (column 1), detection inference results on translated imagery (column 2) and an (unpaired) real image from the Cityscapes dataset, to aid effective translation assessment (column 3).

Figure S3: The Sim10k → Cityscapes adaptation scenario. In all cases the detector model is trained on the Sim10k dataset and evaluated on Cityscapes. We visualise detection inference results on Cityscapes (column 1), inference result on the translated image (column 2) and an (unpaired) real image from the Sim10k dataset, to aid effective translation assessment (column 3).

Figure S4: KITTI → Cityscapes adaptation scenario. In all cases the detector model is trained on KITTI dataset and evaluated on Cityscapes. We visualise detection inference results on Cityscapes (column 1), detection inference results on a translated image (column 2) and an (unpaired) real image from the KITTI dataset, to aid effective translation assessment (column 3).

Det. backbone	G_A	$\mathcal{L}_{G_A}$	$\mathcal{L}_{G_A}$ type	Attention	mAP@[.5:.95]	mAP@.5	mAP@.75	mAP@[.5:.95] small	mAP@[.5:.95] medium	mAP@[.5:.95] large
R-50-C4**			-		23.0	42.7	21.8	2.2	20.8	47.4
	✓		-	✓	23.5	44.4	20.8	2.5	22.3	46.3
	✓	✓	unsupervised	✓	24.1	45.3	23.2	2.6	23.3	47.1
	✓	✓	supervised	✓	24.5	46.7	22.9	2.7	23.4	47.1
R-101-FPN			-		24.3	45.5	21.9	3.8	22.4	46.1
	✓		-	✓	25.2	46.2	23.2	4.0	24.2	46.8
	✓	✓	unsupervised	✓	26.2	49.1	24.4	5.1	25.3	47.1
	✓	✓	supervised	✓	26.0	48.0	24.1	4.7	25.0	46.3
R-50-DC5			-		24.3	44.6	22.7	2.5	21.7	50.4
	✓		-	✓	24.8	45.1	23.4	2.5	23.0	50.1
	✓	✓	unsupervised	✓	25.9	47.2	24.0	2.5	24.1	50.2
	✓	✓	supervised	✓	25.7	46.9	24.5	2.5	23.7	51.6

Table S1: Ablation on method components (Foggy Cityscapes → Cityscapes). We report the effect of ablating method components in columns 2–5, across multiple detector backbone networks. ** denotes R-50-C4 experimental results, also reported in the main paper.

Model	backbone	Weight init.	FPN	Eval. scenario	person	rider	car	truck	bus	train	motor	bike	mAP@.5 ↑
R-50-FPN	ResNet-50	COCO	✓	source	49.1	40.3	46.7	30.3	36.8	24.0	29.1	42.8	37.4
				target oracle	74.9	59.7	60.5	44.0	69.0	58.6	50.1	52.2	58.6
				ours unsupervised	67.0	54.5	58.6	39.7	57.1	44.3	41.3	49.2	51.4
R-50-FPN	ResNet-50	ImageNet	✓	source	46.6	38.6	45.1	20.6	35.6	10.5	29.7	40.3	33.4
				target oracle	74.8	57.8	61.0	42.2	67.0	50.0	50.0	52.1	56.9
				ours unsupervised	69.1	52.9	59.3	36.5	57.3	43.8	46.1	48.6	51.7
R-50-C4**	ResNet-50	ImageNet		source	35.5	38.7	41.5	18.4	32.8	12.5	22.3	33.6	29.4
				target oracle	47.5	51.7	66.9	39.4	56.8	49.0	43.2	47.3	50.2
				ours unsupervised	43.2	50.1	61.7	33.3	48.6	47.8	35.2	42.6	45.3
R-101-C4	ResNet-101	ImageNet		source	35.4	39.3	43.8	22.3	34.9	8.9	23.3	34.1	30.2
				target oracle	46.8	49.6	67.4	40.5	60.6	52.7	42.7	45.4	50.7
				ours unsupervised	43.1	48.2	62.4	35.9	51.7	46.0	36.3	44.3	46.0
R-101-FPN	ResNet-101	ImageNet	✓	source	38.1	43.0	45.2	24.9	37.3	27.4	24.7	37.9	34.8
				source	54.2	57.7	72.7	44.8	59.8	47.3	45.9	48.0	53.8
				ours unsupervised	49.8	54.1	66.7	41.4	52.4	46.3	37.4	37.4	49.1
R-101-DC5	ResNet-101	ImageNet		source	36.5	41.3	46.6	26.8	37.1	16.0	27.2	37.5	33.6
				target oracle	46.3	51.0	67.5	43.8	62.0	52.3	43.7	47.1	51.7
				ours unsupervised	44.0	48.6	62.6	40.2	55.4	42,7	37.3	44.3	46.9
Retina-101-FPN	ResNet-101	ImageNet	✓	source	35.9	38.1	45.5	32.8	29.3	25.0	22.5	29.7	32.4
				target oracle	42.7	48.1	64.5	38.5	50.5	37.6	35.1	39.2	44.7
				ours unsupervised	41.1	46.3	60.5	37.3	47.4	36.4	31.5	37.8	42.3

Table S2: Sensitivity analysis considering backbone model architecture and detector training settings (Foggy Cityscapes → Cityscapes). ** indicates experimental results, reported in the main paper.

architectures. Results are found in Tab. S2. We select the Foggy Cityscapes → Cityscapes adaptation scenario for our analysis. Towards fair comparison with existing work, all results reported in the main paper follow the common experimental setup as described in multiple previous works [11, 29, 30, 43, 71, 73]; i.e. making use of a Faster-RCNN model with a Res-Net-50-C4 backbone. Experimentally, we additionally consider and evaluate a total of six backbones: Res-Net-50-C4, Res-Net-50-FPN, Res-Net-101-C4, Res-Net-101-FPN, Res-Net-101-DC5 and Retina-101-FPN. Futher details regarding the aforementioned architectures are found in [S2]. All models are trained using the hyperparameters and settings reported in Sec. 6.4.

For every experiment we report detection performance on imagery pertaining to *source*, *target oracle*, and *local-global*; which refer to images obtained from Foggy Cityscapes, Cityscapes datasets and images generated by our self-supervised local-global model, respectively. We observe consistent gains, over the baseline *source* evaluation scenario, that range from 9.9–18.3% mAP@.5 under all considered backbones architectures. Models that incorporate the FPN module [S3] (denoted as *-FPN) consistently outperform the baseline backbones (denoted as *-C4), with Retina-101-FPN being an exception. We additionally

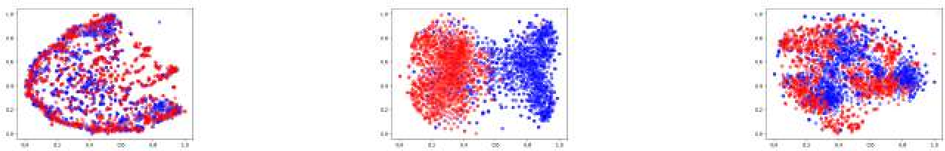

Figure S5: Enlarged version of the main paper feature visuliazation via t-SNE. We randomly sample object features corresponding to salient object and background regions. We compare (a) baseline model without G_A, (b) a model with supervised G_A, (c) a model with self-supervised G_A using the Eq. (5) loss.

observe that pre-training the detector on the COCO dataset [33] improves both source and target domain performance which may be attributed to the fact that the model initialisation has then been optimised for an object detection task.

Our exploration of additional model backbones allows us to evidence scenarios in which our approach, promisingly, brings us to within 2.5% of target oracle performance (*e.g.* Retina-101-FPN). Finally, we note that mAP accuracy increases cf. our main paper results are possible, however we opt to retain the R-50-C4 setting in our manuscript, towards highlighting fair comparisons.

6.3 Comparison with contrastive learning I2I translation methods

We further evaluate the quality of the translated images using standard I2I translation metrics. Tab. S3 reports Frechet Inception Distance [16] (FID) and Kernel Inception Distance [3] (KID), comparing our approach with images generated using CUT [40], FeSeSim [72] and Qs-Attn [19]. We use object labels to explicitly evaluate image quality in regions that contain object instances; by computing the aforementioned metrics exclusively only in those regions. These derivative metrics are denoted FID_{INST} and KID_{INST}, respectively. Interestingly, our method shows improvement in standard I2I translation metrics. Large improvements can be found in our object-region specific metrics, in the case when object labels are available.

We compare our translation results with CUT [40], FeSeSim [72] and Qs-Attn [19] in Fig. S6. It may be observed that while previous methods are successful in transferring the global style and appearance, they often struggle to capture instance-level details and result in poor translation quality in local object areas. By identifying salient object regions, our approach guides the translation task to optimise appearance of object instances and achieve superior image quality in the relevant image regions.

Method	FID ↓	KID ↓	FID_{INST} ↓	KID_{INST} ↓
CUT*† [40] *(ECCV '20)*	0.21	0.84	0.61	2.77
FeSeSim*† [72] *(CVPR '21)*	0.20	0.74	<u>0.51</u>	<u>1.97</u>
Qs-Att.*† [19] *(CVPR '22)*	0.20	0.83	0.55	2.23
Ours - supervised	**0.18**	**0.67**	**0.47**	**1.44**
Ours - local-global †	<u>0.19</u>	<u>0.70</u>	<u>0.51</u>	2.02

Table S3: Comparison of recent contrastive learning based image-to-image translation methods, across image quality metrics, under the Foggy Cityscapes → Cityscapes setting.

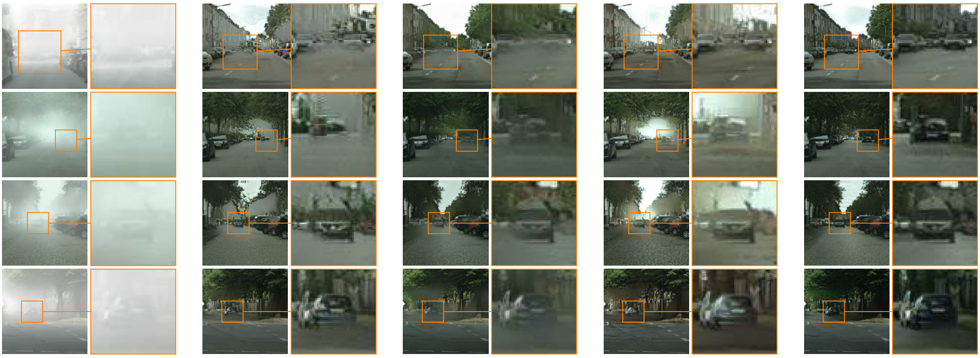

Figure S6: Qualitative comparison with state-of-the-art contrastive learning based I2I methods. We compare against foggy input (Column 1), CUT [40] (Column 2), FeSeSim [72] (Column 3), Qs-Attn [19] (Column 4), Our method (Column 5). Our approach achieves better translation in object regions through the proposed attention driven scheme. Best viewed with digital zoom.

6.4 Implementation details

Detector training details All detectors are trained using identical hyperparameters and settings; we employ Stochastic Gradient Descent (SGD) with base learning rate 0.001, batch size of 4 and weight decay of 5×10^{-4}. Unless otherwise stated, we initialize the models with ImageNet weights and decrease the learning rate after 50000 and 70000 iteration steps with $\gamma = 0.1$, for a total of 100,000 training iterations. Following common protocol, all images are resized such that the smallest side length (i.e. width or height) is 600 pixels both during training and test. All models are implemented using Detectron2 [S7] and PyTorch libraries [S5].

Image translation training details We build our image-to-image (I2I) translation model using a patchwise multi-layer component, similar to [40]. For fair comparison, all models in Tab. 1 and Tab. S3 are trained for a total of 400 epochs using an Adam optimizer [28] with momentum parameters $b1$=0.5, $b2$=0.99 and an initial learning rate $1e{-}5$. The input images are resized such that the smallest size is 600 pixels during training. We perform inference on full resolution images during test.

When training the self-supervised local-global translation model, unless stated otherwise, we follow training settings aligned with [62]. For data augmentation, we apply random horizontal flip, gaussian blur and color jittering related to brightness, contrast, saturation, hue and grayscale. Our local patch generation process follows the approach of [S4]. Namely, a random region is firstly cropped such that it covers at least 60% of the original global image, followed by the aforementioned data augmentation operations. The image is divided into 4×4 grid areas which are randomly shuffled to obtain the final 16 local patches. Finally, we set weights of Eq. (5) to 0.1, 0.4, 0.7, 1.0 for objective terms w_1, w_2, w_3, w_4 respectively, where each term pertains to a different convolutional layer of networks G_A and G_{Am}.

Network Architectures We denote a network convolutional layer that contains f filters, with stride x and a $y \times y$ kernel size to be a cf-sx-ky layer. In this notation convention, $c64$-$s1$-$k3$ denotes a convolutional layer that applies 64 filters with a stride of 1 and kernel size 3×3. Futhermore, we denote a convolutional layer that applies the transposed convolution operation using f filters, a stride of x and kernel size $y \times y$ as uf-sx-ky. Unless stated otherwise, every cf-sx-ky and uf-sx-ky layer is followed by a ReLu [S1] activation function and

InstanceNorm [S6] normalization layer.

We deploy a patchGAN discriminator [S2] D with an architecture that can be denoted by [c64-s2-k4, c128-s2-k4, c256-s2-k4, 512-s1-k4]. Accordingly, we model the feature extractor network E_B using layers [c64-s2-k7, c128-s2-k3, c256-s2-k3, 9·r256-s1-k3] where 9·r256-s1-k3 denotes 9 residual blocks with 2 convolutional layers, each. We implement network G_C as [u128-s2-k7, u64-s2-k3, u27-s1-k7], where the u27-s1-k7 layer is followed by a tanh activation function, without a normalization layer.

For the attention generator G_A we use an architecture denoted by [u128-s2-k7, u64-s2-k3, u10-s1-k7] where the last layer, u10-s1-k7, is followed by a *Softmax* function which generates the attention masks. The supervised model additionally trains two filters c2-s1-k7 in network G_A which produce the object saliency prediction. Our fully self-supervised model follows the same architecture as the supervised model for E_B, G_C and G_A with the only exception being the object saliency filters, c2-s1-k7, in G_A. In the self-supervised case, momentum networks E_{Bm}, G_{Am} follow identical architectural copies of E_B, G_A, respectively. More specifically, we optimize layers [u128-s2-k7, u648-s2-k3, u10-s1-k7] and [$um*$128-s2-k7, $um*$64-s2-k3, $um*$10-s1-k7] together, where $um*$ denotes the corresponding layers of G_{Am}. We additionally optimize layers c256-s2-k3 and $cm*$256-s2-k3 together, pertaining to E_B and E_{Bm} respectively, via Eq. (5) of the main paper. We attach 4 global and 4 local MLP heads to each of these layers to obtain the final representations. The set of MLPs are implemented as a set of linear layers followed by ReLU activation function. All experiments are performed on four NVIDIA V100 GPUs, each with 32GB of RAM.

References

[S1] Abien Fred Agarap. Deep learning using rectified linear units (relu). *ArXiv*, abs/1803.08375, 2018.

[S2] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *2017 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[S3] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, 2017. doi: 10.1109/CVPR.2017.106.

[S4] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6706–6716, 2019.

[S5] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[S6] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. In *ArXiv*, volume abs/1607.08022, 2016.

[S7] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2: A pytorch-based modular object detection library. *Meta AI*, 10:3, 2019.