Supplementary Material

A. Details of the datasets

We include here additional details related to the datasets we used, namely GTA-5 [62], SYNTHIA [65], Cityscapes [20] and ACDC [68]. In particular, in Fig. A.1 we provide example images from each dataset with the corresponding ground-truth segmentation map. Then, in Sec A.1 we discuss which categories we considered, as different datasets have different annotations available. Finally, we provide details about the sequences we designed for the experiments in Sec A.2.

A.1. Classes

In this section, we detail the subset of categories that we considered for our experiments, an important point since some classes are available in some datasets but not in others. For example, lane-marking is available in SYNTHIA but not in GTA-5/Cityscapes/ACDC, and vice-versa the terrain class is available in GTA-5/Cityscapes/ACDC but not in SYNTHIA. We used the 19 classes below from the GTA-5 that are also available in Cityscapes/ACDC: *road, sidewalk, building, fence, pole, light, sign, vegetation, sky, person, car, bicycle, bus, train, motorcycle, wall, terrain, truck, rider*. For GTA-5/Cityscapes/ACDC, we use all of them. For SYNTHIA, we only use the ones highlighted in magenta (the others being unavailable).

A.2. Sequences

In the following, we report the details of the different sequences used in our experiments. This is limited to the SYNTHIA, ACDC, Cityscapes O. and Cityscapes A.W. dataset. In the case of GTA-5, the whole dataset is used for the offline pre-training step; images are randomly sampled from the dataset and the dataset is parsed over several epochs.

SYNTHIA. For clarity, this paragraph uses the sequence IDs as reported in the dataset's directories: *01* is *Highway*, *04* is *Old European Town* and *05* is *New York-like City*. The weather/daylight/seasonal conditions we used are *Summer*, *Spring*, *Fall*, *Winter*, *Dawn*, *Sunset*, *Night*, *Rain*, *Fog*, *Rain-night*, *Winter-night*. To indicate a sub-sequence shift, we use arrows ($\rightarrow$). For each sub-sequence, we indicate the specific environment and the specific weather/daylight condition (*e.g.* , *04/Night*). We use 300 consecutive frames per sub-sequence (to limit the length of the experiments), and build the following sequences – each one totalling $1.5k$ samples:

- *05/Night (300 frames) → 01/Dawn (300 frames) → 01/Winter (300 frames) → 05/Winternight (300 frames) → 04/Softrain (300 frames)*
- *04/Night (300 frames) → 01/Winter (300 frames) → 04/Softrain (300 frames) → 05/Winternight (300 frames) → 05/Fog (300 frames)*
- *01/Winternight (300 frames) → 05/Winternight (300 frames) → 05/Night (300 frames) → 04/Sunset (300 frames) → 04/Winter (300 frames)*
- *05/Winternight (300 frames) → 05/Dawn (300 frames) → 05/Night (300 frames) → 05/Sunset (300 frames) → 01/Winter (300 frames)*
- *05/Softrain (300 frames) → 01/Night (300 frames) → 04/Fog (300 frames) → 05/Winter (300 frames) → 01/Winternight (300 frames)*
- *01/Night (300 frames) → 04/Fog (300 frames) → 01/Fall (300 frames) → 05/Fall (300 frames) → 05/Rain (300 frames)*
- *04/Spring (300 frames) → 05/Winter (300 frames) → 04/Night (300 frames) → 01/Dawn (300 frames) → 04/Rainnight (300 frames)*
- *01/Winter (300 frames) → 04/Sunset (300 frames) → 04/Spring (300 frames) → 01/Spring (300 frames) → 05/Fog (300 frames)*
- *04/Rainnight (300 frames) → 04/Softrain (300 frames) → 05/Winter (300 frames) → 05/Fog (300 frames) → 01/Dawn (300 frames)*

Cityscapes O. We only use frames from the original Cityscapes dataset (without weather variations) for which "fine-grained" annotation is provided (Cityscapes also provides frames for which some "coarse" annotation is provided). Apart from this, we do not perform any cut to the sub-sequences, and build the following sequences:

- *Aachen (174 frames) → Hamburg (248 frames) → Frankfurt (267 frames) → Munster (174 frames)*
- *Jena (119 frames) → Hamburg (248 frames) → Zurich (122 frames) → Hanover (196 frames)*
- *Hamburg (248 frames) → Stuttgart (196 frames) → Tubingen (144 frames) → Darmstadt (85 frames)*
- *Stuttgart (196 frames) → Bochum (96 frames) → Monchengladbach (94 frames) → Bremen (316 frames)*
- *Lindau (59 frames) → Bochum (96 frames) → Aachen (174 frames) → Stuttgart (196 frames)*
- *Monchengladbach (94 frames) → Dusseldorf (221 frames) → Jena (119 frames) → Strasbourg (365 frames)*
- *Jena (119 frames) → Strasbourg (365 frames) → Bochum (96 frames) → Dusseldorf (221 frames)*
- *Strasbourg (365 frames) → Stuttgart (196 frames) → Tubingen (144 frames) → Monchengladbach (94 frames)*
- *Krefeld (99 frames) → Erfurt (109 frames) → Tubingen (144 frames) → Strasbourg (365 frames)*
- *Monchengladbach (94 frames) → Lindau (59 frames) → Aachen (174 frames) → Jena (119 frames)*

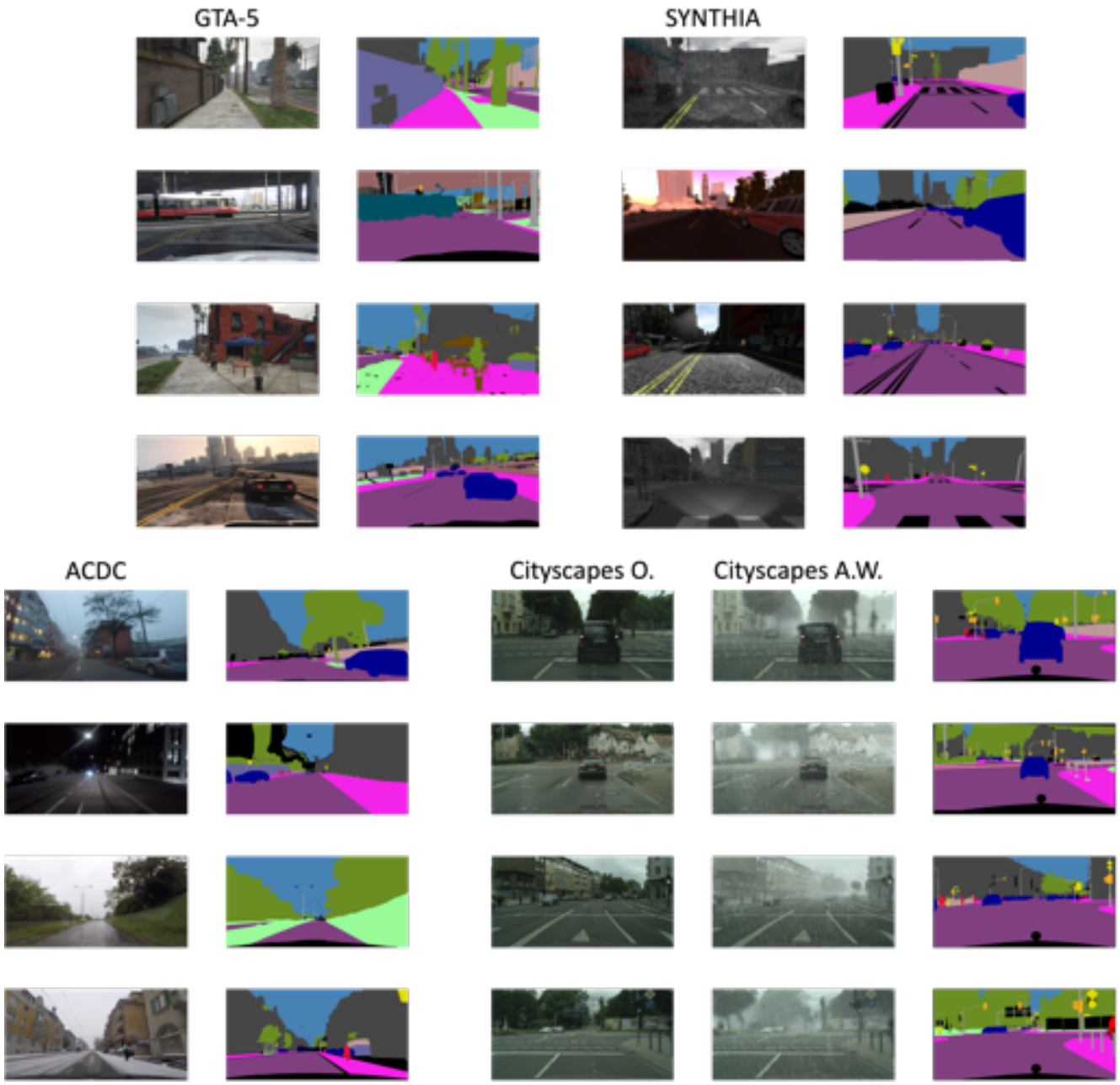

Figure A.1. Sample images of the datasets used for our experiments.

Cityscapes A.W. We define the following sequences by combining sub-sequences (without cut) from the original Cityscapes [20] (Clean), from Cityscapes sequences with artificial *Fog* [67] and with artificial *Rain* [35]. "Clean" indicates that the original sequences are used.

- *Zurich/Clean (122 frames) → Darmstadt/Fog (85 frames) → Dusseldorf/Rain (68 frames) → Jena/Fog (119 frames)*
- *Munster/Rain (30 frames) → Hamburg/Fog (248 frames) → Cologne/Clean (154 frames) → Erfurt/Clean (109 frames)*
- *Bremen/Clean (316 frames) → Stuttgart/Fog (196 frames) → Aachen/Rain (65 frames) → Tubingen/Clean (144 frames)*
- *Dusseldorf/Rain (68 frames) → Darmstadt/Clean (85 frames) → Tubingen/Fog (144 frames) → Bremen/Rain (53 frames)*
- *Bremen/Rain (53 frames) → Krefeld/Clean (99 frames) →*

Lindau/Fog (59 frames) → Bochum/Clean (96 frames)

- *Cologne/Clean (154 frames) → Munster/Rain (30 frames) → Hanover/Fog (196 frames) → Bremen/Clean (316 frames)*
- *Frankfurt/Fog (267 frames) → Erfurt/Rain (59 frames) → Zurich/Clean (122 frames) → Cologne/Clean (154 frames)*
- *Hanover/Clean (196 frames) → Aachen/Fog (174 frames) → Jena/Fog (119 frames) → Munster/Rain (30 frames)*
- *Bremen/Rain (53 frames) → Ulm/Fog (95 frames) → Zurich/Clean (122 frames) → Darmstadt/Fog (85 frames)*
- *Erfurt/Rain (59 frames) → Ulm/Clean (95 frames) → Aachen/Rain (65 frames) → Lindau/Fog (59 frames)*

ACDC. We do not perform any cut to the sub-sequences, and build the following sequences:

- *GP010476/Fog (41 frames) → GP010402/Rain (31 frames) → GP030176/Snow (22 frames) → GP010376/Night (56 frames)*
- *GP010476/Fog (41 frames) → GOPR0351/Night (149 frames) → GOPR0122/Snow (48 frames) → GP020402/Rain (102 frames)*
- *GOPR0402/Rain (83 frames) → GP010376/Night (56 frames) → GP010607/Snow (69 frames) → GOPR0478/Fog (41 frames)*
- *GP040176/Snow (86 frames) → GP020402/Rain (102 frames) → GP020397/Night (44 frames) → GP010476/Fog (41 frames)*
- *GOPR0122/Snow (48 frames) → GP020475/Fog (37 frames) → GOPR0356/Night (50 frames) → GP010402/Rain (31 frames)*

B. Details of the methods

Hyper-parameter selection. We report details related to the choice of hyper-parameters, expanding on Section 5 from the main manuscript. We train our models with a DeepLab-V2 [12] architecture, implemented in PyTorch [60]. We pre-train our models on GTA-5 [62] for 6 epochs, using SGD optimizer with learning rate $\eta = 2.5 \cdot 10^{-4}$, momentum $\alpha = 0.9$ and weight decay $\lambda = 5 * 10^{-4}$. For GPU-memory constraints, we set the batch size to 1. In the following, we report the hyper-parameters associated with each method.

- **N-BN:** BN momentum $\alpha = 0.1$
- **C-BN:** BN momentum $\alpha = 0.1$
- **N-TENT:** learning rate $\eta = 1.0$
- **C-TENT:** learning rate $\eta = 0.01$
- **C-TENT-SR:** learning rate $\eta = 0.01$, source regularizer weight $\gamma = 1.0$
- **Class-R-TENT:** learning rate $\eta = 0.1$, $K = 1$, $\psi = 1.0$
- **Oracle-R-TENT:** learning rate $\eta = 1.0$
- **N-PL:** learning rate $\eta = 10^{-4}$
- **C-PL:** learning rate $\eta = 10^{-4}$
- **C-PL-SR:** learning rate $\eta = 10^{-4}$, source regularizer weight $\gamma = 2.0$
- **Class-R-PL:** learning rate $\eta = 10^{-4}$, $K = 1$, $\psi = 1.0$
- **Oracle-R-PL:** learning rate $\eta = 10^{-4}$

The hyper-parameters were cross-validated on SYNTHIA sequences, and kept unchanged for ACDC, Cityscapes A.W. and Cityscapes O. sequences.

Domain randomization (DR). To perform domain randomization (DR [77]), we modify the aspect of training samples by applying K different, random image transformations before feeding each sample to the model. Referring to notation in Table 2 in the main paper, we validate $K = 2, 3, 4$ for DR↑, DR↑↑ and DR↑↑↑ respectively. In the following, we report the different transformations we rely on during training and we refer the reader to PIL [1–3] for a more detailed documentation.

- *Identity:* the image remains unchanged.
- *Brightness:* the brightness of the image is perturbed, with intensity in the range $[0.2; 1.8]$.
- *Color:* the color of the image is perturbed, with intensity in the range $[0.2; 1.8]$.
- *Contrast:* the contrast of the image is perturbed, with intensity in the range $[0.2; 1.8]$.
- *RGB perturbations:* a random scalar in the range $[0; 120]$ is added to each of the RGB channels.
- *RGB-to-gray:* the image is converted to grayscale.

C. Additional results

We provide in this section additional results, to extend the main ones included in the manuscript. To provide a roadmap, in Sec. C.1 we analyze the effect of BN momentum η on the N-BN method; in Sec. C.2 we show adaptation results obtained when starting from the ERM source model; in Sec. C.3 we analyze how the results vary in the iterative methods N-PL and N-TENT when increasing the number of adaptation iterations; in Sec. C.4 we provide additional ACDC results, for sequences where only the urban environment change, but the weather/daylight condition is fixed; and finally, in Sec. C.5 we provide more continual learning curves such as the one shown in Fig. 3 in the main paper.

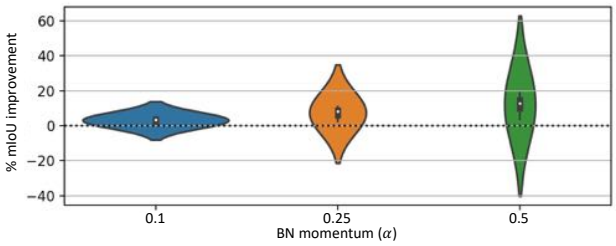

Figure C.2. SYNTHIA results for N-BN – varying BN momentum (α).

C.1. BN adaptation

In Figure C.2 we showcase the distribution of N-BN results for the SYNTHIA sequences, when we vary the BN momentum ($\alpha = 0.1, 0.25, 0.5$ in blue, orange and green, respectively). The general trend is that increasing the momentum α – that means, increasing the impact of the target sample's statistics when mixing those with the source statistics – leads to higher average results at the price of a significantly larger spread. These results indicate that the algorithm will perform significantly better on some sequences, and significantly worse in others. Since this approach is versatile and could be applied in tandem with any other method (for example, one could perform continual adaptation with a reset mechanism, and also adapt BN statistics with increased BN momentum on each sample), we believe that a more thorough understanding of its behavior on the proposed task represents an interesting research direction.

C.2. Adapting from ERM v.s. adapting from DR

In Table C.1 we compare the adaptation results on ACDC between the models started from ERM with the models started from DR↑↑. We can observe that none of the models started from ERM achieves the performance of the baseline DR↑↑ – apart the Oracle-R, which starting from ERM performs on par with the DR↑↑ NA baseline. This confirms the crucial importance of the initial $\mathcal{M}_{\theta_0}$. A second observation we can make is that while the numbers are lower for models that start the adaptation from ERM (first column), the overall improvement trend is consistent in general; this emphasizes that the conclusions made concerning the different adaptation strategies hold even if we change the initial $\mathcal{M}_{\theta_0}$.

C.3. Adaptation iterations for PL and TENT

In Table C.2 we compare N-PL and N-TENT results as we increase the number of iterations. For example, in the case of N-PL we iterate several times between pseudo-labeling and updating the model. We can observe that, in general, increasing the number of iterations yields better results for N-PL. In the case of N-TENT, in the multi-iteration case, results can be significantly improved by reducing the learning rate (except in the case of Cityscapes O.). Naturally, this improvement comes with an increased computational cost, which can be prohibitive according to specific applications – *e.g.* , autonomous driving.

Comparing adaptation from ERM and DR

	Pre-trained model	
Method	**ERM**	**DR↑↑**
No adaptation	29.5 ±2.5	33.6 ±2.5
Style transfer		
N-ST (NN)	27.4 ±2.2	31.9 ±2.3
N-ST (rand)	25.5 ±1.6	31.1 ±1.7
Naive adapt.		
N-BN	30.2 ±2.6	34.4 ±2.5
N-PL	30.4 ±2.6	34.6 ±2.5
N-TENT	31.5 ±2.7	35.3 ±2.7
CL adapt.		
C-BN	31.7 ±2.2	35.9 ±2.3
C-PL	27.8 ±3.4	29.7 ±3.6
C-TENT	31.2 ±2.8	34.5 ±3.4
CL+SR adapt.		
C-PL-SR	31.3 ±2.8	34.5 ±2.7
C-TENT-SR	31.1 ±2.9	35.6 ±2.8
Adaptive-reset adapt.		
Class-N-PL	32.4 ±2.3	36.3 ±2.3
Class-N-TENT	32.1 ±2.3	36.0 ±2.3
Oracle-reset adapt.		
Oracle-N-PL	33.6 ±2.4	37.5 ±2.3
Oracle-N-TENT	33.6 ±2.4	37.2 ±2.3

Table C.1. Comparison between models adapted starting from ERM or DR↑↑ pre-training. Results reported in mIoU.

C.4. Sequences with multiple cities and fixed weather/daylight conditions

We report in Table C.3 results associated with ACDC sequences where the condition (*Fog*, *Night*, *Rain*, *Snow*) is fixed, and only the urban environment change. For each condition, results are averaged over the following sequences:

Fog:

- *GP020475* → *GOPR0478* → *GOPR0476* → *GP010476*
- *GOPR0477* → *GP020478* → *GP010476* → *GOPR0475*
- *GP020475* → *GOPR0476* → *GOPR0478* → *GP010476*
- *GOPR0477* → *GOPR0479* → *GOPR0476* → *GP020475*

Varying the number of iterations for N-PL and N-TENT

Method	Adapt iter.	Learn. rate	Sequence type: SYNTHIA	ACDC	Cityscapes A.W.	Cityscapes O.
N-PL	1	0.0001	+3.5% ±1.0	+2.9% ±0.6	+2.4% ±1.0	+1.4% ±0.2
N-PL	3	0.0001	+7.8% ±2.7	+6.5% ±1.7	+5.2% ±2.6	+2.4% ±0.4
N-PL	5	0.0001	+9.5% ±3.7	+8.4% ±2.7	+6.5% ±3.7	+2.2% ±0.6
N-TENT	1	1.0	+8.5% ±3.1	+4.9% ±2.0	+3.1% ±3.6	−1.2% ±0.7
N-TENT	3	1.0	+8.4% ±4.2	+4.0% ±3.1	+3.4% ±5.3	−3.0% ±1.0
N-TENT	5	1.0	+6.8% ±4.6	+2.9% ±3.5	+2.8% ±5.9	−4.5% ±1.2
N-TENT	1	0.1	+4.0% ±1.1	+3.1% ±0.7	+2.6% ±1.2	+1.4% ±0.2
N-TENT	3	0.1	+8.9% ±2.9	+6.9% ±1.9	+5.4% ±3.0	+2.1% ±0.4
N-TENT	5	0.1	+10.7% ±3.8	+8.5% ±2.9	+6.4% ±4.2	+1.5% ±0.6

Table C.2. Results (relative performance gain in %) obtained on N-PL and N-Tent when increasing the number of training iterations.

Night

- *GOPR0351 → GP010376 → GOPR0356 → GP020397*
- *GP020397 → GOPR0356 → GOPR0376 → GOPR0351*
- *GP010397 → GP010376 → GOPR0351 → GOPR0356*
- *GOPR0356 → GOPR0351 → GOPR0376 → GP010376*

Rain

- *GOPR0400 → GP020400 → GP020402 → GOPR0402*
- *GP010400 → GP020402 → GOPR0400 → GOPR0402*
- *GP010400 → GOPR0400 → GP010402 → GP020400*
- *GOPR0402 → GOPR0400 → GP010400 → GP010402*

Snow

- *GP010607 → GOPR0604 → GOPR0606 → GOPR0122*
- *GOPR0606 → GP010122 → GP050176 → GOPR0607*
- *GOPR0607 → GP010122 → GOPR0604 → GP030176*
- *GP010607 → GP030176 → GOPR0604 → GOPR0606*

Discussion. When we analyze the results in the Table C.3, on **Fog** and **Snow** sequences we can observe a behavior similar to the one that we have observed for the ACDC dataset obtained with the multi-environment multi-condition sequences (Table 1). On the other hand, **Night** and **Rain** sequences represent an interesting case study, with some discrepancies. For example, we can observe significant improvements when adapting with Style Transfer (ST) to Night sequences due to the fact that the ST can effectively increase the overall brightness and contrast of the image (see Fig. C.8), making it easier for the model to process such samples. Furthermore, note that due to a strong domain gap between GTA-5 (containing mainly day images) and images all over these sequences (all night images), the NA baseline yields to a poor performance even with DR↑↑. The con-

ACDC results (multi environment / single condition)

	Sequence type: Fog	Night	Rain	Snow
No adapt. (NA)	41.4 ±1.5	15.6 ±0.9	41.9 ±0.2	38.9 ±0.8
Method	**Improvements**			
Style transfer				
N-ST (NN)	−8.8% ±2.1	−3.3% ±0.5	−6.4% ±0.6	−2.8% ±0.6
N-ST (rand)	−13.9% ±1.3	+20.6% ±1.6	−9.9% ±0.5	−10.3% ±0.5
Naive adaptation				
N-BN	+2.7% ±0.8	+4.5% ±0.3	+1.6% ±0.2	+2.1% ±0.4
N-PL	+3.5% ±1.1	+5.1% ±0.3	+1.9% ±0.2	+2.7% ±0.4
N-TENT	+9.1% ±3.7	+5.5% ±0.7	+1.2% ±0.5	+7.1% ±1.2
CL adaptation				
C-BN	+8.4% ±5.1	+23.7% ±2.1	+0.0% ±0.8	+6.8% ±2.1
C-PL	+0.8% ±6.2	−41.2% ±21.1	−15.8% ±2.3	−12.2% ±6.3
C-TENT	+8.7% ±5.7	−8.1% ±2.0	−3.0% ±1.0	+4.6% ±2.1
CL+SR adaptation				
C-PL-SR	+8.2% ±2.6	−10.8% ±1.7	−2.0% ±0.9	+4.3% ±2.0
C-TENT-SR	+6.1% ±3.0	+0.0% ±3.2	+0.6% ±0.1	+3.5% ±1.3
Adaptive-reset adaptation				
Class-N-PL	+9.8% ±5.6	+25.0% ±2.2	+0.2% ±0.9	+7.3% ±2.0
Class-N-TENT	+9.4% ±6.0	+24.6% ±2.1	−0.4% ±0.9	+6.8% ±2.3
Oracle-reset adaptation				
Oracle-N-PL	+13.0% ±5.9	+32.2% ±2.7	+2.7% ±0.8	+10.4% ±1.9
Oracle-N-TENT	+13.6% ±6.4	+31.1% ±2.9	+1.6% ±0.8	+10.1% ±2.1

Table C.3. Results (relative performance gain in %) on ACDC sequences built with multiple cities but fixed weather/daylight conditions.

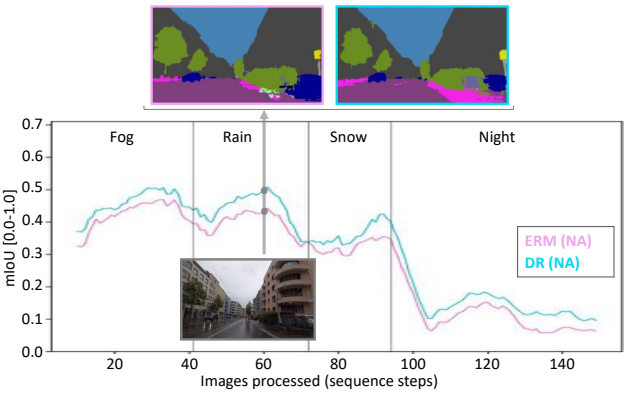

Figure C.3. **ERM *vs* DR** Performance evolution of ERM and DR non-adapted (NA) models, in pink and light blue, respectively – for one ACDC [65] sequence. In the plot, it can be observed that DR brings consistent improvements with respect to ERM. Top: qualitative comparison between predictions of the two methods when processing the reported image; notice the *sidewalk* on the bottom-right corner.

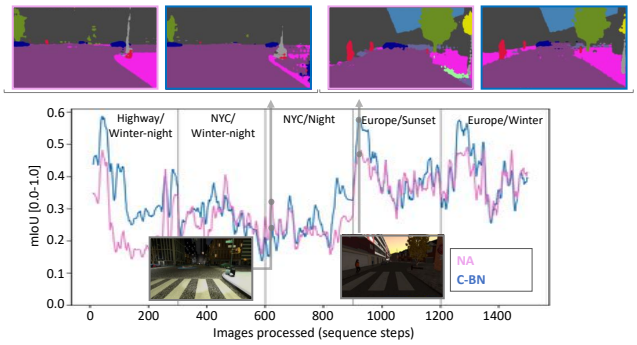

Figure C.4. **C-BN *vs* NA.** Performance evolution of a non-adapted model (NA) and a model whose BN statistics is updated continuously [53], in pink and blue, respectively – for one SYNTHIA [65] sequence. In the plot, it can be observed that C-BN generally brings consistent improvements with respect to NA. Top: qualitative comparison between predictions of the two methods when processing the reported image (on the left, we report failure cases of C-BN w.r.t. NA, from the limited number of frames in which C-BN underperforms.).

tinual learning methods C-PL and C-TENT fail in this condition, and SR does not carry the same improvements than it did in other setups. The best performing strategies are C-BN and the reset methods. For the **Rain** sequences, we observe a similar behavior to the one already observed for Cityscapes O. (Table 1). Starting from a strong DR↑↑ baseline model, none of the continual methods, excluding the Oracle-R approach, brings significant improvements, and the gain of Oracle-R over the baseline is relatively modest. Surprisingly, there is a significant improvement for the Fog sequence, a condition for which the NA model performs as well as for Rain. Properly understanding different condition-dependent behaviors represents an interesting research question for real-world applications.

C.5. Qualitative plots

We finally report additional qualitative plots, showing the performance evolution as in Figure 3 in the main paper. We report them in Figures C.3—C.6, extending Figure 3 from the main manuscript. Apart from models reported in Figure C.3, results reported in all other figures assume DR pre-training. We summarize the content of each figure below, and report the details in the different captions.

- *Figure C.3*: We compare models with and without DR [77] pre-training, in *light blue* and *pink*, respectively.
- *Figure C.4*: We compare models trained with and without BN [38] statistics online adaptation [53], in *blue* and *pink*, respectively.
- *Figure C.5*: We compare models trained via N-PL, C-PL and Class-R-PL, in *blue*, *orange* and *green*, respectively.
- *Figure C.6*: We compare models trained via N-PL, C-PL and C-PL-SR, in *blue*, *orange* and *red*, respectively.

Figure C.7 *(top)* provides a qualitative view on improvements led by adapting the model with C-TENT-SR *(top-right)* with respect to the non-adapted baseline *(top-middle)*. In this specific example, the improvement in classifying the sidewalk's pixels is very significant. Figure C.7 *(bottom)* provides qualitative evidence of "catastrophic forgetting", which we discussed about in the main manuscript. When the model is trained continuously and without regularization, it can forget classes if it does not encounter them for a while. In this specific example, the C-TENT model has forgotten the pedestrian class almost completely *(bottom-middle)*. By regularizing with a term that optimizes the cross-entropy loss on source samples, the C-TENT-SR model is continuously exposed to the source classes; hence, it does not forget about them *(bottom-right)*.

D. Style transfer

One of the most popular approaches for unsupervised domain adaptation is to apply a photorealistic transformation to the source images (usually the training set) so that they resemble the *style* of the target images (usually the test or deployment data) [19, 52, 63, 73, 92]. However these methods assume that, even if unlabeled, both the source and target image distributions are known at train time. In this work, we are proposing a benchmark where test images are unknown a priori and we receive them as a continuous stream. Thus, we can not apply these techniques directly. On the other hand, a related line of work has focused on performing single-image style transfer: Given a *content* image and a *style* image, transform the content image to mimic

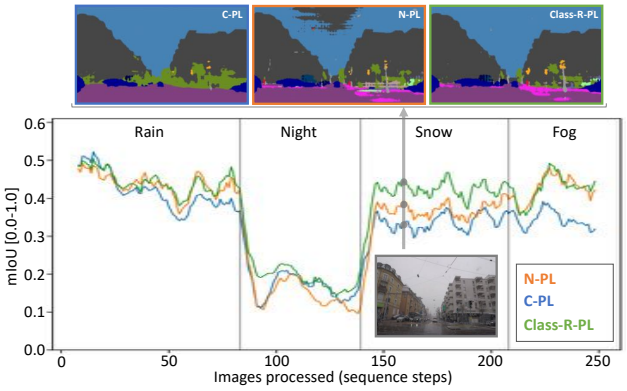

Figure C.5. **Naive *vs* Continual *vs* Reset.** Performance evolution of Naive, "Vanilla" continual, and Class-reset versions of PL (N-PL, C-PL and Class-R-PL, is orange, blue and green, respectively) – for one ACDC [68] sequence. In the plot, it can be observed that learning continuously without precautions results in sub-optimal performance, and that reset can allow maintaining a performance close to the naive counterpart in some parts of the sequence, while significantly improving over it in others. Top: qualitative comparison between predictions of different methods when processing the reported image; C-PL has catastrophically forgotten *sidewalks*, *pedestrians* and *poles* (best in color zooming in).

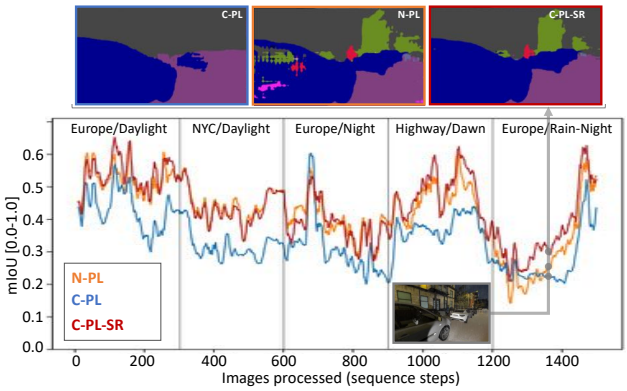

Figure C.6. **Naive *vs* Continual *vs* Source regularized.** Performance evolution of Naive, "Vanilla" continual, and Source-regularized versions of PL (N-PL, C-PL and Class-R-PL, is orange, blue and red, respectively) – for one SYNTHIA [65] sequence. In the plot, it can be observed that learning continuously without precautions results in sub-optimal performance, and that source regularization helps mitigating such negative impact, leading to performance often Naive. Top: qualitative comparison between predictions of different methods when processing the reported image; C-PL has catastrophically forgotten *pedestrians* and *vegetation* (best in color zooming in).

the high-level appearance of the style image. Several works in this area have focused on artistic style transfer, where they provide images with painting styles e.g. [28,37,40,91]. However, when applying these methods between two images they create artifacts that yield unrealistic results. This motivated other works to focus on photorealistic style transfer [4,44,51,59,98]. Although these works were generally motivated from an aesthetics perspective, we note photorealistic style transfer can be used to perform online unsupervised domain adaptation, where for each image received at test time we apply style transfer as a pre-processing step to mimic the appearance of *some* image from the training set. In our work, we use the method described in [98].

D.1. Experimental details

We use the public implementation[8] from [98] with default parameters and stylize at all modules (encoder, decoder and skip connections). We do not use segmentation masks since the content image is unlabeled and using our prediction as a segmentation mask might induce unwanted artifacts. In our experiments, we apply style transfer to every test (or validation) image independently as a pre-processing step. For every content image (test or validation) we choose a corresponding style image (from the training set) to apply the style transfer. We compare two strategies:

Random selection: We select images from the training set uniformly, regardless of the appearance and semantic content of images. This selection method is indeed fast, however, it might pair style and content images which are very different and lead to somewhat artificial image styles.

Nearest neighbour selection: In this case, the idea is to match each content image with the closest style image. We compare images using the cosine similarity between the features extracted after applying a CNN encoder. We note that we can pre-compute the features of the training images offline. Moreover, we use the same encoder used for style transfer as feature extractor for further efficiency.

Regardless of the selection method, style transfer is a rather expensive procedure which takes of the order of 1-2 seconds per image on a GPU NVIDIA V100, depending on the image size. Therefore, in order to apply it in a real-time scenario, further work should be dedicated into speeding-up the style transfer process. In Figure C.8 we provide illustrative samples of content, style and stylized images for each dataset and sampling method.

E. Code and Streamlit web app

Our code to replicate the experiments provided in this work is attached to the submission, see the files in `2419_code.zip`.

We report in Fig. E.9 a screenshot of the Web application we will release to explore results and models (the file used to generate it is `streamlit_app.py`). On the left, one can select the specific model and sequence; on the right, the results are reported; in the middle, the ground truth, image and predicted masks are reported – the selection can be made via the slider on the bottom-left. We find this app

[8] https://github.com/clovaai/WCT2

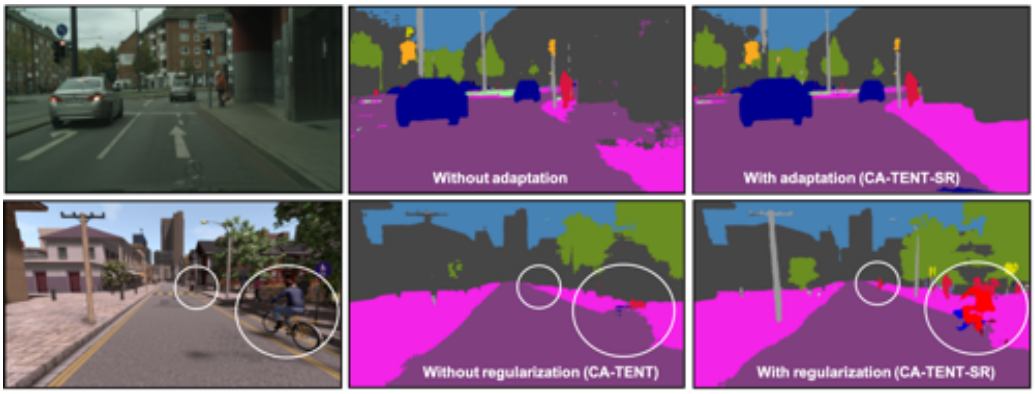

Figure C.7. *Top:* We compare the predictions of a non-adapted model *(top-middle)* and a model adapted via CA-TENT-SR *(top-right)*. This image shows how the non-adapted model struggles in classifying the sidewalk's pixels, while the adapted model improves in this regard. *Bottom:* We compare the predictions of models trained via two different versions of the TENT [87] algorithm, when provided with an image *(left)* CA-TENT (*bottom-middle*) and CA-TENT-SR *(bottom-right)*. This comparison shows how models trained continuously but without regularization can catastrophically forget some classes – in this case, the pedestrian one.

Figure C.8. Sample images generated with the style transfer baseline from each dataset. The original *content* image is associated either a random or a nearest neighbour *style* image from the GTA-5 dataset. Then we apply a style transfer method on the content-style pair to obtain a *stylized* image. Note how when choosing style images randomly we can obtain somewhat unrealistic stylized images while choosing them with a simple nearest neighbour search helps mitigate this issue.

very useful for research on semantic segmentation, where exploring qualitative results is as important as assessing final performance (*e.g.*, final mIoU values). The same app can be used to generate the plots shown in Fig. C.3–C.6

F. Limitations

We conclude by highlighting the limitations of ideas and methods detailed in this work.

For what concerns the OASIS benchmark we introduced (our core contribution), we tried, as much as possible, to mimic conditions that one may face when deploying a machine learning system in the real world – in particular, testing on samples/sequences significantly different from the ones on which the models have been trained and validated. Yet, it is important to remember that the real world can expose our models to a variety of conditions significantly broader than the ones a benchmark can contain. Thus, it is important to avoid having a false sense of security before deploying a system in the real world; for example, if we consider an outdoor robot, there may be combinations of weather/visual conditions and urban environments, not considered in the benchmark, that might significantly alter its performance.

For what concerns the methodology, it is important to

Figure E.9. Screenshot of our Streamlit web app.

gain familiarity with the failure cases of unsupervised domain adaptation. As we tried to convey with the experiments reported in Table C.1, the starting point is extremely important to foster good adaptation results; if the pre-trained model is significantly under-performing in some conditions, domain adaptation can hardly improve on such performance (in some cases, it can even deteriorate the performance even more). Finally, concerning the implications of continual learning – and, more specifically, of continual unsupervised adaptation – catastrophic forgetting is a severely limiting factor. While in Sec. 4.3 we have proposed a family of methods based on a reset mechanism; this is a very simple heuristic, far from solving a very broad research problem. Still, we hope that raising these limitations will encourage the community to consider the problem and devise more advanced solutions to tackle it.