Self-supervised Augmentation Consistency for Adapting Semantic Segmentation
– Supplemental Material –

Nikita Araslanov[1] Stefan Roth[1,2]
[1]Department of Computer Science, TU Darmstadt [2] hessian.AI

A. Overview

In this appendix, we first provide further training and implementation details of our framework. We then take a closer look at the accuracy of long-tail classes, before and after adaptation. Next, we discuss our strategy for hyper-parameter selection and perform a sensitivity analysis. We also evaluate our framework using another segmentation architecture, FCN8s [98]. Finally, we discuss the limitations of the current evaluation protocol and propose a revision based on the best practices in the field at large.

B. Further Technical Details

Photometric noise. Recall that our framework uses random Gaussian smoothing, greyscaling and colour jittering to implement the photometric noise. We re-use the parameters for these operations from the MoCo-v2 framework [93]. In detail, the kernel radius for the Gaussian blur is sampled uniformly from the range $[0.1, 2.0]$. Note that this does not correspond to the actual filter size.[1] The colour jitter, applied with probability 0.5, implements a perturbation of the image brightness, contrast and saturation with a factor sampled uniformly from $[0.6, 1.4]$, while the hue factor is sampled uniformly at random in the range of $[0.9, 1.1]$. We convert a target image to its greyscale version with probability 0.2. Fig. 6 demonstrates an example implementation of this procedure in Python.

Constraint-free data augmentation. Similarly to the multi-scale cropping of the target images, we scale the source images randomly with a factor sampled uniformly from $[0.5, 1.0]$ prior to cropping. However, we do not enforce the semantic consistency for the source data, since the ground truth of the source images is available. For both the target and source images we also use random horizontal flipping. We additionally experimented with moderate rotation (both with and without semantic consistency), but did not observe a significant effect on the mean accuracy.

[1]The Pillow Library [94] internally converts the radius r to the box length as $L = \sqrt{3 * r^2 + 1}$.

```python
import random
import PIL
import torchvision.transforms as tf
import torchvision.transforms.functional as F

# Load the image
image = PIL.Image.open(...)

# Gaussian blur
# with a randomly sampled radius
radius = random.uniform(.1,2.)
gaussian = PIL.ImageFilter.GaussianBlur(radius)
image = image.filter(gaussian)

# Colour jitter
# with probability 0.5
if 0.5 > random.random():
  jitter = tf.ColorJitter(brightness=0.4,
                          contrast=0.4,
                          saturation=0.4,
                          hue=0.1)
  image = jitter(image)

# Convert to greyscale
# with probability 0.2
if 0.2 > random.random():
  image = F.to_grayscale(image)
```

Figure 6. **Python implementation of the photometric noise.**

Training schedule. Our framework typically needs $150-200$K iterations in total (*i.e.* including the source-only pre-training) until convergence, as determined on a random subset of 500 images from the training set (see our discussion in Appendix D below). This varies slightly depending on the backbone and the source data used. This schedule translates to approximately 3 days of training with standard GPUs (*e.g.*, Titan X Pascal with 12 GB memory) for both VGG-16 and ResNet-101 backbones. Recall that we used 4 GPUs for our ResNet version of the framework, hence its training time is comparable to the VGG variant, which uses only 2 GPUs. All our experiments use a constant learning rate for simplicity, but more advanced schedules, such as cyclical learning rates [35], the cosine schedule [93, 95] or ramp-ups [40], may further improve the accuracy of our framework.

CBT	IS	FL	road	sidew	build	wall	fence	pole	light	sign	veg	terr	sky	pers	ride	car	truck	bus	train	moto	bicy	mIoU
			88.1	41.0	85.7	30.8	30.6	33.1	37.0	22.9	86.6	36.8	90.7	67.1	27.1	86.8	34.4	30.4	8.5	7.5	0.0	44.5
		✓	89.4	52.3	86.0	**34.0**	32.6	**38.5**	43.3	30.6	85.2	30.9	88.5	66.7	28.0	85.7	35.6	39.6	0.0	6.6	0.0	46.0
	✓		**90.0**	47.1	85.6	31.3	24.9	32.3	38.9	28.2	**87.3**	**39.8**	89.4	**67.7**	28.6	**88.1**	40.1	50.0	7.3	9.9	2.2	46.8
✓			89.3	39.0	85.1	33.2	26.1	32.4	41.8	25.2	86.3	27.4	**90.4**	66.4	28.2	87.5	32.9	45.4	11.0	7.6	0.0	45.0
	✓	✓	89.3	52.6	86.0	33.4	30.0	38.0	44.9	34.3	86.9	35.3	88.0	65.4	27.3	86.2	37.6	44.0	20.9	9.6	6.5	48.2
✓		✓	89.3	52.2	86.1	34.2	31.5	37.0	43.4	36.3	85.2	30.7	86.6	66.2	**30.3**	85.3	36.2	43.9	**29.2**	6.8	8.6	48.4
✓	✓		89.7	45.1	85.6	29.6	28.3	31.7	41.9	27.5	87.2	37.4	89.8	66.9	29.2	87.5	37.3	31.6	24.7	11.9	20.2	47.5
✓	✓	✓	**90.0**	**53.1**	**86.2**	33.8	**32.7**	38.2	**46.0**	**40.3**	84.2	26.4	88.4	65.8	28.0	85.6	**40.6**	**52.9**	17.3	**13.7**	**23.8**	**49.9**

Table 5. **Per-class IoU (%)** on Cityscapes *val* using a VGG-16 backbone in the GTA5 → Cityscapes setting. We study three components in more detail: class-based thresholding (CBT), importance sampling (IS) and the focal loss (FL). The mIoU of the settings in the last four rows are reproduced from the main text. Here, we elaborate on the per-class accuracy in a broader context of the supplementary experiments in the first four rows.

C. Additional Experiments

C.1. A closer look at long-tail adaptation

Recall that our framework features three components to attune the adaptation process to the long-tail classes: class-based thresholding (CBT), importance sampling (IS) and the focal loss (FL), which we summarily refer to as the *long-tail components* in the following. Disabling the long-tail components individually is equivalent to setting $\beta \to 0$ for CBT, using uniform sampling of the target images instead of IS or assigning λ to 0 for the FL. Here, we extend our ablation study of the GTA5 → Cityscapes setup with VGG-16 (*cf.* Table 4 from the main text) and experiment with different combinations of the long-tail components. Table 5 details the per-class accuracy of the possible compositions.

We observe that the ubiquitous classes – "road", "building", "vegetation", "sky", "person" and "car" – are hardly affected; it is primarily the long-tail categories that change in accuracy. Furthermore, our long-tail components are mutually complementary. The mean IoU improves when one of the components is active, from 44.5% to up to 46.8%. It is boosted further with two of the components enabled to 48.4%, and reaches its maximum for our model, 49.9%, when all three components are in place.

We further identify the following tentative patterns. FL tends to improve classes "wall", "fence" and "pole". CBT increases the accuracy of the "traffic light" category (which has high image frequency and occupies only a few pixels), but also rare classes, such as "rider", "bus" and "train" benefit from CBT, especially in conjunction with IS. IS also enhances the mask quality of the classes "bicycle" and "motorcycle". Nevertheless, we urge caution against interpreting the results for each class in isolation, despite such widespread practice in the literature. Today's semantic segmentation models do not possess the notion of an 'ambiguous' class prediction and each pixel receives a meaningful label. By the pigeon's hole principle, this implies that the changes in the IoU of one class have an immediate effect on the IoU of the other classes. Therefore, the benefits of individual framework components have to be understood in the context of their aggregated effect on multiple classes, *e.g.* using the mean IoU. For instance, consider the class "train" for which IS appears to also decrease the IoU: while CBT together with FL achieve 29.2% IoU, adding IS decreases the IoU to 17.3%. However, the IoU of other classes increases (*e.g.*, "motorcycle", "bicycle"), as does the mean IoU. Furthermore, only few classes reach their maximum accuracy when we enable all three long-tail components. Yet, it is the setting with the best *accuracy trade-off* between the individual classes, *i.e.* with the highest mean IoU. Overall, the long-tail components improve our framework by 5.4% mean IoU combined, a substantial margin.

C.2. Hyperparameter search and sensitivity

To select ζ and β, we first experimented with a few reasonable choices ($\zeta \in (0.7, 0.8)$, $\beta \in (0.0001, 0.01)$)[2] using a more lightweight backbone (MobileNetV2 [97]). To measure performance, we use the mean IoU on the validation set (500 images from Cityscapes *train*, as in the main text).

Here, we study our framework's sensitivity to the particular choice of ζ and β. To make the results comparable to our previous experiments, we use VGG-16 and report the mean IoU on Cityscapes *val* in Table 7. We observe moderate deviation of the IoU w.r.t. ζ. A more tangible drop in accuracy with $\beta = 0.01$ is expected, as it leads to low-confidence predictions, which are likely to be inaccurate, to be included into the pseudo label. We note that while a sub-optimal choice of these hyperparameters leads to inferior results (with a standard deviation of $\pm 1.4\%$ mIoU), even the weakest model with $\zeta = 0.8$ and $\beta = 0.01$ did not fail to considerably improve over the baseline (by 8.5% IoU, *cf.* Table 2 in the main text).

[2]While ζ may seem more interpretable (the maximum confidence threshold), a reasonable range for β can be derived from χ_c for the long-tail classes, which is simply the fraction of pixels these classes tend to occupy in the image (see Eq. 3).

Method	road	sidew	build	wall	fence	pole	light	sign	veg	terr	sky	pers	ride	car	truck	bus	train	moto	bicy	mIoU
GTA5 → Cityscapes																				
Baseline (ours)	76.7	28.2	74.4	12.7	19.0	27.2	28.7	12.2	77.0	18.0	70.6	54.8	20.6	79.6	19.0	19.2	20.6	27.9	11.2	36.7 (37.1)
SAC-FCN (ours)	86.3	45.6	84.4	30.3	27.1	24.8	42.8	35.2	86.9	39.7	88.0	62.3	32.1	84.1	28.4	43.7	31.9	29.4	45.8	49.9 (49.9)
SYNTHIA → Cityscapes																				
Baseline (ours)	50.7	23.8	60.9	1.8	0.1	27.7	10.5	15.7	60.1	—	72.4	50.1	16.0	66.5	—	13.7	—	8.5	26.8	31.6 (34.4)
SAC-FCN (ours)	74.7	34.2	81.4	19.8	1.9	27.2	34.8	27.2	80.0	—	86.3	61.5	20.8	82.5	—	31.2	—	32.0	53.9	46.8 (49.1)

Table 6. **Per-class IoU (%)** on Cityscapes *val* using VGG-16 with FCN8s. For reference, the numbers in parentheses in the last column report the mean IoU of the DeepLabv2 architecture (*cf.* Tables 2 and 3 from the main text).

$\downarrow \zeta$ / $\beta \rightarrow$	0.0001	0.001	0.01
0.7	47.9	48.8	46.7
0.75	48.6	49.9	46.3
0.8	48.2	49.8	45.6

Table 7. **Mean IoU (%) on GTA5 → Cityscapes (val) with varying ζ and β.** Our framework maintains strong accuracy under different settings of ζ and β. Even with a poor choice (*e.g.*, $\zeta = 0.8$, $\beta = 0.01$), it fares well w.r.t. the state of the art and outperforms many previous works (*cf.* Table 2 from the main text).

C.3. VGG-16 with FCN8s

A number of previous works (*e.g.*, [55, 77, 80]) used the FCN8s [98] architecture with VGG-16, as opposed to DeepLabv2 [10], adopted in other works (*e.g.*, [39, 70]) and ours. Such architecture exchange appears to have been dismissed in previous work as minor, which used only one of the architectures in the experiments. However, the segmentation architecture alone may contribute to the observed differences in accuracy of the methods and, more critically, to the improvements otherwise attributed to other aspects of the approach. To facilitate such transparency in our work, we replace DeepLabv2 with its FCN8s counterpart in our framework (with the VGG-16 backbone) and repeat the adaptation experiments from Sec. 4, *i.e.* using two source domains, GTA5 and SYNTHIA, and Cityscapes as the target domain. We keep the values of the hyperparameters the same, with an exception of the learning rate, which we increase by a factor of 2 to 5×10^4. Table 6 reports the results of the adaptation, which clearly show that our framework generalises well to other segmentation architectures. Despite the FCN8s baseline model (source-only loss with ABN) achieving a slightly inferior accuracy compared to DeepLabv2 (*e.g.*, 31.6% *vs.* 34.4% IoU for SYNTHIA → Cityscapes), our self-supervised training still attains a remarkably high accuracy, 46.8% IoU (*vs.* 49.1% with DeepLabv2). This is substantially higher than the previous best method using FCN8s with the VGG-16 backbone, SA-I2I [55]: +3.4% on GTA5 → Cityscapes and +5.3% on SYNTHIA → Cityscapes.

D. Towards Best-practice Evaluation

The current strategy to evaluate domain adaptation (DA) methods for semantic segmentation is to use the ground truth of 500 randomly selected images from the Cityscapes *train* split for model selection and to report the final model accuracy on the 500 Cityscapes *val* images [47]. In this work, we adhered to this procedure to enable a fair comparison to previous work. However, this evaluation approach is evidently in discord with the established best practice in machine learning and with the benchmarking practice on Cityscapes [18], in particular.

The test set is holdout data to be used only for an unbiased performance assessment (*e.g.*, segmentation accuracy) of the final model [96]. While it is conceivable to consult the test set for verifying a number of model variants, such access cannot be unrestrained. This is infeasible to ensure when the test set annotation is public, as is the case with Cityscapes *val*, however. Benchmark websites traditionally enable a restricted access to the test annotation through impartial submission policies (*e.g.*, limited number of submissions per time window and user), and Cityscapes officially provides one.[3]

We, therefore, suggest a simple revision of the evaluation protocol for evaluating future DA methods. As before, we use Cityscapes *train* as the training data for the target domain, naturally without the ground truth. For model selection, however, we use Cityscapes *val* images with the ground-truth labels. The holdout test set for reporting the final segmentation accuracy after adaptation becomes Cityscapes *test*, with the results obtained via submitting the predicted segmentation masks to the official Cityscapes benchmark server.

An additional advantage of this strategy is a clear interpretation of the final accuracy in the context of fully supervised methods that routinely use the same evaluation setup. Also note that Cityscapes *val* contains images from different cities than Cityscapes *train* (which are also different from Cityscapes *test*). Therefore, it is more suitable for detecting cases of model overfitting on particularities of the city, since the validation set was previously a subset of the

[3] https://www.cityscapes-dataset.com

Method	road	sidew	build	wall	fence	pole	light	sign	veg	terr	sky	pers	ride	car	truck	bus	train	moto	bicy	mIoU
GTA5 → Cityscapes																				
SAC-FCN (ours)	87.5	45.2	85.0	29.2	26.4	23.3	44.2	32.0	88.3	52.6	91.2	65.2	35.0	86.0	24.4	32.8	31.4	36.9	40.5	50.4 (49.9)
SAC-VGG (ours)	91.5	53.9	86.6	34.1	31.5	36.8	47.2	36.9	85.1	38.0	91.1	68.7	31.9	87.4	31.0	46.7	22.6	24.2	24.0	51.0 (49.9)
SAC-ResNet (ours)	91.8	54.3	87.4	36.2	30.2	43.7	49.7	42.1	89.3	54.3	90.5	71.8	34.9	89.8	38.8	47.3	24.9	38.3	43.8	55.7 (53.8)
SYNTHIA → Cityscapes																				
SAC-FCN (ours)	66.9	25.9	80.8	12.1	2.0	24.4	37.1	27.5	78.8	—	88.9	63.9	25.0	84.7	—	27.4	—	36.9	50.2	45.8 (46.8)
SAC-VGG (ours)	70.4	29.7	83.6	11.6	1.8	34.2	41.2	29.2	81.0	—	87.1	67.9	25.4	75.9	—	34.3	—	42.5	57.5	48.3 (49.1)
SAC-ResNet (ours)	87.4	41.0	85.5	17.5	2.6	40.5	44.7	34.4	87.9	—	91.2	68.0	31.0	89.3	—	33.2	—	38.6	49.9	52.7 (52.6)
Fully supervised (Cityscapes)																				
DeepLab-ResNet [10]	97.9	81.3	90.4	48.8	47.4	49.6	57.9	67.3	91.9	69.4	94.2	79.8	59.8	93.7	56.5	67.5	57.5	57.7	68.8	70.4
FCN-VGG [98]	97.4	78.4	89.2	34.9	44.2	47.4	60.1	65.0	91.4	69.3	93.9	77.1	51.4	92.6	35.3	48.6	46.5	51.6	66.8	65.3

Table 8. **Per-class IoU (%)** on Cityscapes *test*. In the last column, the numbers in parentheses report the mean IoU on Cityscapes *val* from the previous evaluation scheme (*cf.* Tables 2 and 3 from the main text) for reference. SAC-FCN denotes our VGG-based model with FCN8s [98] from Appendix C.3.

training images.

For future reference, we evaluate our framework (both the DeepLabv2 and FCN8s variants) in the proposed setup and report the results in Table 8. To ease the comparison, we juxtapose our validation results reported in the main text (from Table 6 for FCN8s).[4] As we did not finetune our method to Cityscapes *val* following the previous evaluation protocol, we expect the test accuracy on Cityscapes *test* to be on a par with our previously reported accuracy on Cityscapes *val*. The results in Table 8 clearly confirm this expectation: the segmentation accuracy on Cityscapes *test* is comparable to the accuracy on Cityscapes *val* (SYNTHIA → Cityscapes) or even tangibly higher (GTA5 → Cityscapes). We remark that the remaining accuracy gap to the fully supervised model is still considerable (70.4% *vs*. 55.7% IoU achieved by our best DeepLabv2 model and 65.3% *vs*. 51.0% IoU compared to our best FCN8s variant), which invites further effort from the research community.

We hope that future UDA methods for semantic segmentation will follow suit in reporting the results on Cityscapes *test*. Owing to the regulated access to the test set, we believe this setting to offer more transparency and fairness to the benchmarking process, and will successfully drive the progress of UDA for semantic segmentation, as it has done in the past for the fully supervised methods.

[4] To our best knowledge, no previous work published their results in this evaluation setting before.

References

[93] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv:2003.04297 [cs.CV]*, 2020. 1

[94] Alex Clark. Pillow (PIL fork) documentation, 2015. 1

[95] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *ICLR*, 2017. 1

[96] Brian D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996. 3

[97] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *CVPR*, pages 4510–4520, 2018. 2

[98] Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(4):640–651, 2017. 1, 3, 4