Multi-scale and Cross-scale Contrastive Learning for Semantic Segmentation

Supplementary Material

A Selecting loss hyperparameters

The proposed loss (Eq. (5)) requires selecting certain hypeparameters, namely: the number of feature scales, the choice of cross-scale pairs, the per scale and overall weights of contrastive losses. Our results are obtained with minimal model or dataset-specific tuning of those parameters. Specifically, for **all models and datasets** we set both weights of Eq.(5) to 0.1 and use 2 scale-pairs (s4-s32, s4-s16) based on the results of the ablation in Tables 1(a) and 1(c) of the main paper. We further tested two different per-scale weight and cross-scale pair choices using a single model (HRNet) on Cityscapes (Table 1.b) and adopt per-scale weights as a decreasing function of the output stride. Finally, we tested two different alternatives regarding the position of the loss when using the UPerNet architecture, where the loss can be applied either on the FPN outputs or the directly over the backbones features. For Cityscapes the optimal choice is the latter while on ADE20K it is the former (Table 1(a)) and we adopt these choices for all other experiements on each dataset when using UPerNet. Thus, with minimal tuning our approach is effective while potentially further model- or dataset-specific tuning can boost performance even more.

Table 1: Ablation on (a) the position of application of the multi-scale and cross-scale losses for models using the UPerNet architecture and (b) on values of weights w_s of the multi-scale loss of Eq. (4).

(a)

Network	Loss position	Dataset	mIoU (ss)
UPerNet R101	Backbone	CTS	**79.1**
UPerNet R101	FPN	CTS	78.4
UPerNet Swin-S	Backbone	CTS	**81.7**
UPerNet Swin-S	FPN	CTS	80.9
UPerNet Swin-S	Backbone	ADE20K	47.9
UPerNet Swin-S	FPN	ADE20K	**49.0**

(b)

w_s	mIoU (ss)
1.0 1.0 1.0 1.0	81.8
1.0 0.7 0.4 0.1	**82.2**

B Additional ablations

We report additional ablations regarding the effect of using longer training schedules and the importance of using the sampling process described in Section 3.3. As can be seen our method benefits by a longer training schedule and a bigger batch size while staying ahead of the baseline in all cases (Table 2(b)). Further, as shown in Table 2(a), our use of anchor sampling is necessary to allow an extension of contrastive losses to multiple scales as memory consumption exceeds our utilized hardware's capacity ($4 \times$ 24GB-GPUs). Further we find that even with a single contrastive loss term ($\mathcal{L}_c$) our choice to perform anchor sampling results in better performance than using all available anchors in the batch which is equivalent to obtaining a number of anchors per class (denoted by **K**) according to the class distribution p_{data}, which is imbalanced.

Table 2: (a) Comparison with alternative sampling options (40K steps, with a batch size of 8 and using 4). We denote the number of samples per class by **K**. (b) Ablation of training schedules/batch sizes. All results are on **Cityscapes val** using single scale evaluation.

(a)

Model	Scales	Scale Pairs	Loss	Sampling	**K**	mIoU	Mem/GPU (GB)
HRNet	1	-	$\mathcal{L}_c$		$\sim p_{data}$	79.4	14.2
HRNet	1	-	$\mathcal{L}_c$	✓	Sec. 3.3	80.2	7.4
HRNet	4	2	$\mathcal{L}_{cms} + \mathcal{L}_{ccs}$		$\sim p_{data}$	-	OOM
HRNet	4	2	$\mathcal{L}_{cms} + \mathcal{L}_{ccs}$	✓	Sec. 3.3	81.5	9.7

(b)

Network		Settings	mIoU		
Model	Loss	Batch	40K	80K	120K
HRNet	CE	8	79.1	79.7	80.5
HRNet	+ours	8	**81.5**	**81.7**	**81.6**
HRNet	CE	12	-	-	81.0
HRNet	+ours	12	-	-	**82.2**

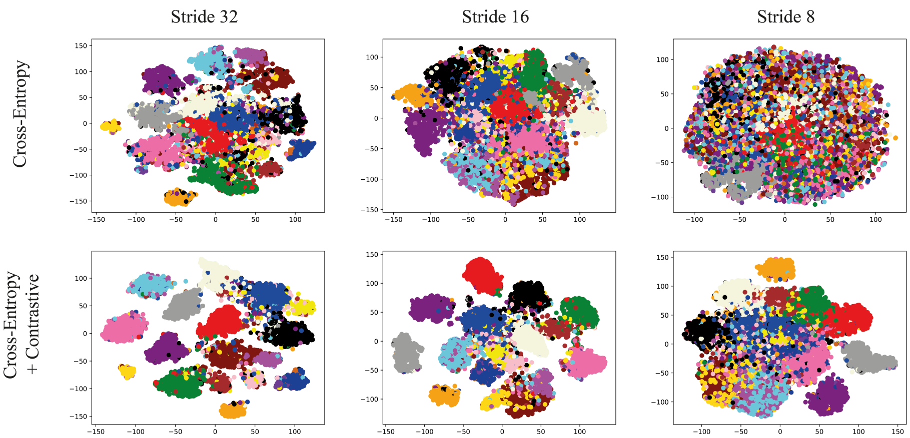

Fig. 1: TSNE [2] visualisation of the feature spaces of UPerNet with ResNet-101 backbone, on Cityscapes, trained without (top) and with (bottom) our proposed contrastive loss. Color indicates each sample's class.

C Additional comparisons on ADE20K

We provide more comparisons between our results using UPerNet with Swin backbones and other state-of-the-art transformer models, on ADE20K. Notably, our result using Swin-B outperforms other competitive models despite having close to a third of the parameters in comparison to Segmenter [3] and SETR [5].

Table 3: Additional results and comparisons with SOTA on **ADE20K val**.

Network				Loss	mIoU	
Model	**Backbone**	**#Params** (M)	**Source**		ss/ms	Improvement
UPerNet	Swin-B†	121	[1]	CE	50.1/51.6	
UPerNet	Swin-B†	121	-	ours	**51.3/52.2**	(+1.2/ + 0.6)
UPerNet	Swin-L†	234	[1]	CE	52.0/53.5	
UPerNet	Swin-L†	234	-	ours	**52.9**/53.3	(+0.9/ − 0.2)
SegFormer	MiT-B5	84	[4]	CE	51.1/51.8	
Segmenter	ViT-L/16†	307	[3]	CE	50.7/52.2	
SETR	T-Large†	310	[5]	CE	48.6/50.3	

D Qualitative results

We provide qualitative results of models trained with our proposed loss on ADE20K (Fig. 2), Cityscapes (Fig. 3) and CaDIS (Fig. 4). We also compare the feature spaces of UPerNet with ResNet-101 backbone, without and with our contrastive loss (Fig. 1).

E Training and testing settings

In Tables 5, 4, 6 and 7 we provide the settings used for our experiments. We closely follow each baseline's implementation details found in its official code publication. Regarding testing, when multi-scale (flipping and scaling) inference is used, the scaling factors used are $0.5, 0.75, 1.25, 1.5, 1.75$ on **ADE20K** and $0.5, 0.75, 1.25, 1.5, 1.75, 2.0$ on **Cityscapes-test** and **Pascal-Context**.

Table 4: Training settings on **Cityscapes**.

Network		Settings					
Model	**Backbone**	**crop**	**lr**	**decay**	w_d	**Batch/steps**	**optim**
HRNet	HR48v2	512×1024	10^{-2}	poly	5×10^{-5}	$12/120K$	SGD
OCRNet	HR48v2	512×1024	10^{-2}	poly	5×10^{-5}	$12/120K$	SGD
DeepLabv3	R101	512×1024	10^{-2}	poly	5×10^{-5}	$12/120K$	SGD
UPerNet	R101	512×1024	10^{-2}	poly	5×10^{-5}	$12/120K$	SGD
UPerNet	Swin-T	512×1024	6×10^{-5}	linear	10^{-2}	$8/120K$	ADAMW
UPerNet	Swin-S	512×1024	6×10^{-5}	linear	10^{-2}	$8/120K$	ADAMW
UPerNet	Swin-B	512×1024	6×10^{-5}	linear	10^{-2}	$8/120K$	ADAMW

Table 5: Training settings on **ADE20K**.

Network		Settings					
Model	**Backbone**	**crop**	**lr**	**decay**	w_d	**Batch/steps**	**optim**
OCRNet	HR48v2	512×512	10^{-2}	poly	10^{-4}	$16/160K$	SGD
DeepLabv3	R101	512×512	10^{-2}	poly	10^{-4}	$16/160K$	SGD
UPerNet	R101	512×512	10^{-2}	poly	10^{-4}	$16/160K$	SGD
UPerNet	Swin-T	512×512	6×10^{-5}	linear	10^{-2}	$16/160K$	ADAMW
UPerNet	Swin-S	512×512	6×10^{-5}	linear	10^{-2}	$16/160K$	ADAMW
UPerNet	Swin-B	512×512	6×10^{-5}	linear	10^{-2}	$16/160K$	ADAMW
UPerNet	Swin-L	640×640	6×10^{-5}	linear	10^{-2}	$16/160K$	ADAMW

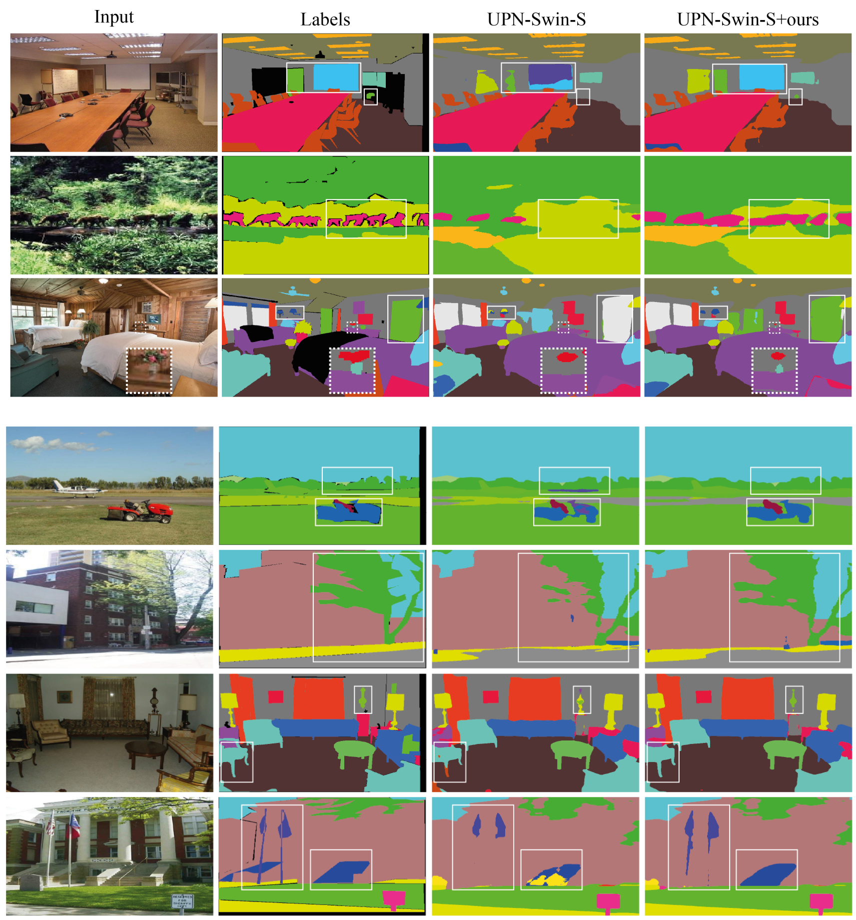

Fig. 2: **Qualitative comparisons on ADE20K val**: we compare UPerNet-Swin-S trained with only CE to the same model trained with also our multi- and cross-scale losses. White bounding boxes indicate some examples where our model performs better in cases where the foreground class is difficult to distinguish from the background (2nd, 5th row) or when it recognizes and segments smaller/thinner objects missed by the baseline (1st, 3rd, 6th, 7th)
.

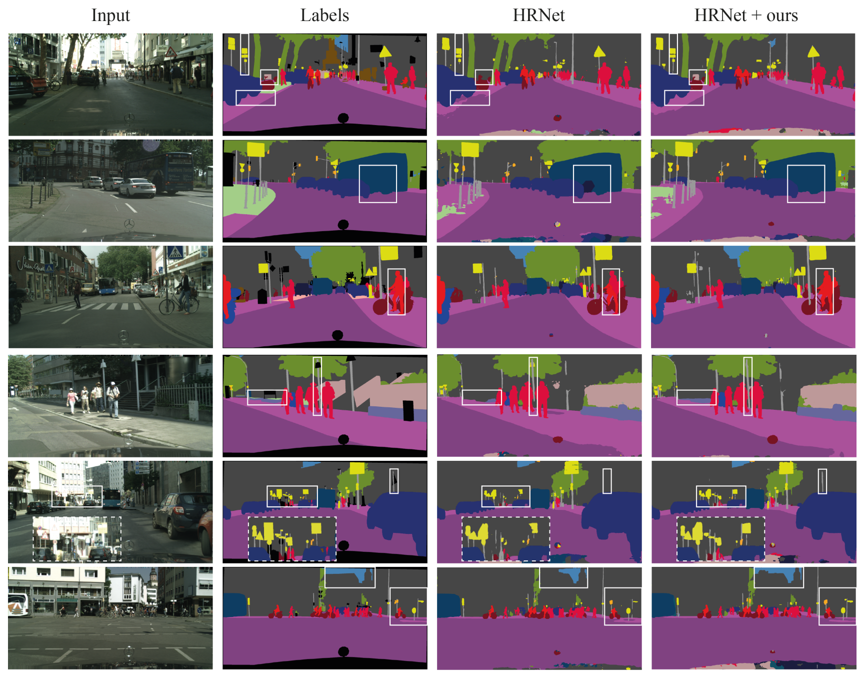

Fig. 3: **Qualitative results on Cityscapes validation set**: we present more qualitative results, comparing HRNet trained with CE to HRNet it trained with our multi- and cross-scale losses. White bounding boxes, outline some of the differences between the two models. Notably, the 2^{nd} and 3^{rd} rows depict cases where the baseline, misclassifies local segments of an object instance, namely a bus is partially recognized as "truck" and a bike rider is partially recognized as a simple pedestrian (i.e "person" in the dataset classes). Our model does not produce these inconsistencies in those cases, showcasing better ability to consider local-global interactions in recognizing and delineating an object instance. Other rows demonstrate examples where the model trained with our loss performs better than the baseline, at delineating small objects with fine details such as traffic signs or poles.

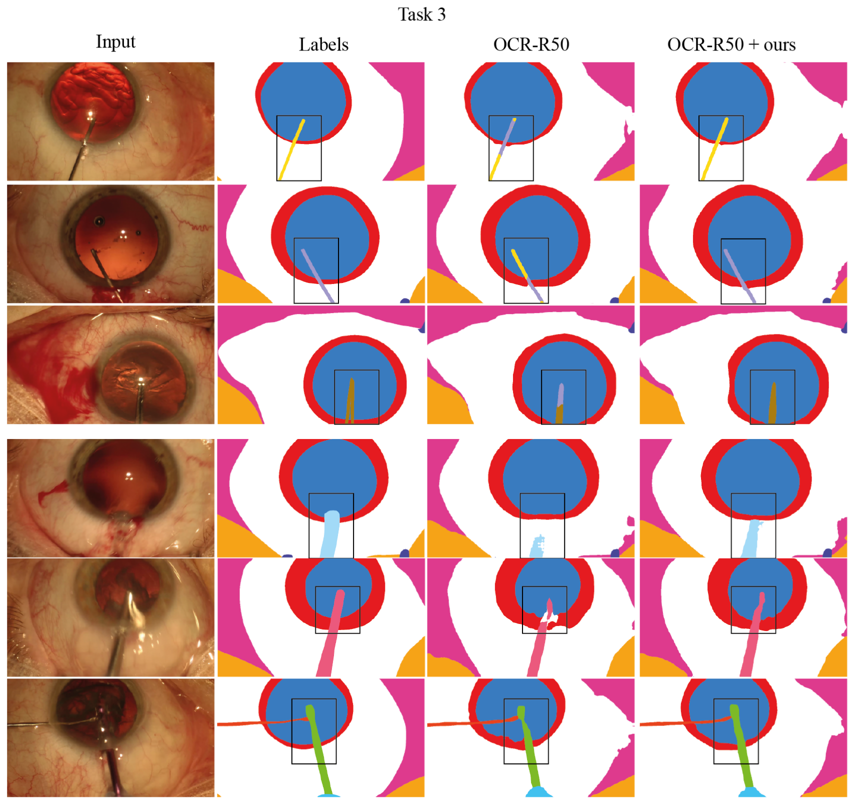

Fig. 4: **Qualitative results on CaDIS validation set for task** 3: We present a visual comparison of the baseline and the result of combining it with our multi- and cross-scale losses. Rows 1-3 demonstrate falsely recognized instrument classes by the baseline whereas our result accurately segments and classifies the tools. Notably, all 3 cases, correspond to tools that have very similar appearance but should be discriminated in task 3, that requires fine grained segmentation and classification. Further, rows 4-6 demonstrate, a barely humanly visible translucent tool and two blurry and specular images with tools, respectively. In all three cases our model achieves clearly more accurate delineation of the tools than the baseline, under challenging conditions.

Table 6: Training settings on **Pascal-Context**.

Network		Settings					
Model	**Backbone**	**crop**	**lr**	**decay**	w_d	**Batch/steps**	**optim**
OCRNet	HR48v2	512×512	10^{-3}	poly	10^{-4}	$16/160K$	SGD
HRNet	HR48v2	512×512	10^{-3}	poly	10^{-4}	$16/160K$	SGD

Table 7: Training settings on **CaDIS**.

Network		Settings					
Model	**Backbone**	**crop**	**lr**	**decay**	w_d	**Batch/steps**	**optim**
OCRNet	R50	540×960	10^{-4}	exp	-	$8/20K$	ADAM

References

1. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 10012–10022 (October 2021)
2. Van der Maaten, L., Hinton, G.: Visualizing data using t-sne. Journal of machine learning research **9**(11) (2008)
3. Strudel, R., Garcia, R., Laptev, I., Schmid, C.: Segmenter: Transformer for semantic segmentation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 7262–7272 (October 2021)
4. Xie, E., Wang, W., Yu, Z., Anandkumar, A., Alvarez, J.M., Luo, P.: Segformer: Simple and efficient design for semantic segmentation with transformers. In: Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems (2021), `https://openreview.net/forum?id=OG18MI5TRL`
5. Zheng, S., Lu, J., Zhao, H., Zhu, X., Luo, Z., Wang, Y., Fu, Y., Feng, J., Xiang, T., Torr, P.H., Zhang, L.: Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6881–6890 (June 2021)