A. Preliminaries of Transformer

We provide details of the Multi-Head Attention (MSA) block we mentioned in Section 4.1 for exhaustivity. ViT [4] inherits the exact $\mathbf{qkv}$ self-attention proposed in [9]. The query, key and value $\mathbf{q}, \mathbf{k}, \mathbf{v} \in \mathbf{R}^{N \times D_h}$ are linearly projected from the input tokens $\mathbf{X} \in \mathbb{R}^{N \times D}$. Each output token is a weighted sum over all values $\mathbf{v}$ in the sequence, where the weights A_{ij} are based on the pairwise similarity between two elements in the sequence with respect to their $\mathbf{q}_i, \mathbf{k}_j$ representations. Self-attention can be formulated as:

$$[\mathbf{q}, \mathbf{k}, \mathbf{v}] = \mathbf{X}\mathbf{U}_{qkv} \qquad \mathbf{U}_{qkv} \in \mathbb{R}^{D \times 3D_h}, \tag{1}$$

$$A = \mathrm{softmax}\left(\mathbf{q}\mathbf{k}^\top / \sqrt{D_h}\right) \qquad A \in \mathbb{R}^{N \times N}, \tag{2}$$

$$\mathrm{SA}(\mathbf{X}) = A\mathbf{v}. \tag{3}$$

Multi-head self-attention (MSA) is an extension of the self-attention where k SA heads are applied to the input sequence in parallel. It returns a linear projection of the concatenated outputs of the SAs:

$$\mathrm{MSA}(\mathbf{X}) = \left[\mathrm{SA}_1(\mathbf{X}), \mathrm{SA}_2(\mathbf{X}), \ldots, \mathrm{SA}_k(\mathbf{X})\right] \mathbf{U}_{msa}, \tag{4}$$

where $\mathbf{U}_{msa} \in \mathbb{R}^{(kD_h) \times D}$.

B. More Implementation Details

Algorithm 1 Code of "painting" in a PyTorch-like style.

```
# y: (B, N, K)        # The token logits.
# Q: (B, nhw, H, W)   # The affinity map Q. n = 9.

# get neighbors for each cell
y = rar(y, "B N K -> B K H W")
nb = im2col(y, kernel_size=3, padding=1)
nb = rar(nb, "B (K n) (H W) -> B H W n K")

# produce output logits map
Q = rar(Q, "B (n h w) H W -> B H W (h w) n")
out = mm(Q, nb)
out = rar(out, "B H W (h w) K -> B (Hh) (Ww) K")
```

`rar`: rearrange of dimensions; `mm`: matrix multiplication.

In Algorithm 1, We provide the psuedo code of the "painting" process in Section 4.3, i.e., producing the final logits map from the token logits and the predicted pixel-token affinity map $\mathbf{Q}$.

C. Extended Ablation Study

In this section, we provide more ablation results as a supplementary of Section 5.3 in the main paper. All the experiments are conducted using half of the training schedule (i.e., 80k for ADE20K and 40k for Cityscapes) unless otherwise specified. We report median value of three runs.

Output Strides Thanks to the class-agnostic region design, we can attain output segmentation of *arbitrary resolutions* with negligible overheads. Here we investigate the effect of output stride, which is determined by the h, w mentioned in Section 4.2. We report results in Table 1.

arch.	$h \times w$	stride	GFLOPs	#params.	ADE20K	City.
ViT-Ti/16	-	-	3.8G	5.7M	38.76	72.02
RegProxy-Ti/16	1×1	$\times 16$	3.9G	5.8M	39.30	73.11
	2×2	$\times 8$	+0.0G	+0.0M	40.23	74.96
	4×4	$\times 4$	+0.0G	+0.0M	**40.76**	75.08
	8×8	$\times 2$	+0.1G	+0.1M	40.51	**75.31**
	16×16	$\times 1$	+0.4G	+0.4M	**41.01**	**75.15**
ViT-S/16	-	-	14.9G	22.0M	45.04	75.39
RegProxy-S/16	1×1	$\times 16$	15.1G	22.2M	45.81	76.71
	2×2	$\times 8$	+0.0G	+0.0M	46.91	78.48
	4×4	$\times 4$	+0.1G	+0.1M	47.18	**78.71**
	8×8	$\times 2$	+0.2G	+0.2M	**47.20**	78.37
	16×16	$\times 1$	+0.8G	+0.9M	**47.34**	**78.81**
RegProxy-R26+S/32	1×1	$\times 32$	16.2G	36.3M	45.40	74.66
	2×2	$\times 16$	+0.0G	+0.0M	46.97	77.56
	4×4	$\times 8$	+0.1G	+0.1M	**48.11**	**78.18**
	8×8	$\times 4$	+0.2G	+0.2M	**47.79**	78.05
	16×16	$\times 2$	+0.3G	+0.9M	47.42	**78.34**

Table 1. **Comparison of different output strides.** We report single scale results on ADE20K and Cityscapes. The GFLOPs are evaluated on 512×512 crops. In gray are the linear baselines. We **bold** the top-2 entries for each model.

Despite the almost free cost of attaining high resolution results, it is not always the best choice to set a large (h, w). Part of the reason is that the performance upper bound tends to saturate as the prediction gets finer. We also hypothesize that a too large (h, w) makes the model harder to train. In the main paper, we report results with (h, w) set to $(4, 4)$ to align the output stride with common segmentation models [2, 10]. Still, increasing (h, w) is a good choice which generally improves the performance with negligible cost.

The 3×3 Conv The 3×3 depth-wise convolution in the affinity head was initially introduced to fuse local information for region geometrics prediction. We investigate its effect in Table 2. We find it improves the performance on ADE20K, while has no significant effect on Cityscapes. We also notice a normal convolution (with group of 1) yields similar results with the depth-wise one. To sum up, the 3×3 depth-wise convolution in the affinity head improves the performance, however it is not a determinative component. Use early transformer layers alone can also achieve considerable performances. This is reasonable since it is only used for region geometrics prediction, while is not involved in the actual context modeling.

Pre-training We study the effect of different ViT pre-trainings. We initialize our model using three settings: 1) Random initialization; 2) DeiT [8] pre-training on ImageNet-1k; 3) AugReg [6] pre-training on ImageNet-21k following recent Segmenter [7]. On RegProxy-S/16, we also report results using DINO [1] self-supervised pre-training. Table 3 summarizes the results. Similar to many recent works [5, 7, 11], RegProxy benefits from pre-training

arch.	3×3 conv.	dpt.[†]	GFLOPs	#params.	ADE20K	City.
RegProxy-Ti/16			3.9G	5.7M	40.01	75.04
	✓		+0.3G	+0.4M	40.72	74.96
	✓	✓	+0.0G	+0.1M	**40.76**	**75.08**
RegProxy-S/16			15.0G	22.1M	46.78	78.63
	✓		+1.3G	+1.3M	46.98	78.68
	✓	✓	+0.1G	+0.2M	**47.18**	**78.71**

[†] Whether to use depth-wise convolution [3].

Table 2. **Effect of the 3×3 conv.**

on large-scale image dataset, while a random initialization will lead to a dramatic performance drop. However, the RegProxy model without pre-training still performs better than its counterpart reported in [7] (18.83 mIoU *vs.* 4.42 mIoU on ADE20K, both using ViT-S/16 backbone).

arch.	pre-train	IN-21k	self-sup.	ADE20K	Cityscapes
RegProxy-Ti/16	none	-	-	13.96	42.58
	DeiT [8]			39.42	74.56
	ViT [6]	✓		**40.76**	**75.08**
Segmenter [7]	none	-	-	4.42	-
RegProxy-S/16	none	-	-	18.83	49.36
	DeiT [8]			45.73	77.69
	DINO [1]		✓	42.21	77.57
	ViT [6]	✓		**47.18**	**78.71**

Table 3. **Performances using different pre-training.**

D. More Experimental Results

Region Semantics Regularization We find adding explicit regularization with respect to the semantical homogeneity of the learned regions will harm the performance. The approach is to minimize the L_2-norm of the region category histogram[1]:

$$\text{hist}\left(\mathbf{s}\right) = L_1\left(\sum_{\mathbf{p}:\mathbf{s}\in\mathcal{N}_\mathbf{p}} q_\mathbf{s}(\mathbf{p})\cdot\text{onehot}\left(\hat{y}(\mathbf{p})\right)\right), \quad (5)$$

where $\hat{y}(\mathbf{p})$ is the ground truth of pixel $\mathbf{p}$. The results are shown in Table 4. The models with explicit regularization perform worse with evidential gaps.

arch.	w/ regularization	ADE20K	Cityscapes
RegProxy-Ti/16		**40.76**	**75.08**
	✓	40.66	74.45
RegProxy-S/16		**47.18**	**78.71**
	✓	46.38	78.02

Table 4. **Effect of explicit regularization on region semantics.**

Multi-Level Features As a common technique, using multi-level feature for token logits prediction also improves the performance of our RegProxy models. Specifically, we feed the concatenated tokens features (of layer $L/2$, $3L/4$ and L) to the linear classifier, instead of using the output tokens of the last layer. The results are reported in Table 5. However, the improvements are marginal, hence in the main paper, we report results without features concatenation.

arch.	multi-level feat.	ADE20K	Cityscapes
RegProxy-Ti/16		40.76	75.08
	✓	**40.98**	**75.54**
RegProxy-S/16		47.18	78.71
	✓	**47.31**	**78.88**

Table 5. **Effect of predicting on concatenated token features.**

E. More Qualitative Results

Geometrics of the Leaned Regions As a supplementary of Section 5.3, in Figure 1, we select a few tokens that have been classified to specific classes, and visualize their corresponding regions. Note the region geometrics is class-agnostic. The heat map is acquired by stacking the probabilistic region descriptions. The learned regions capture fine-grained boundaries, even for small/thin classes such as `pole` and `traffic light` and complicated classes such as `person`. For tokens that locate at the deep inside of the semantics areas (e.g., No.13 in Figure 1), their corresponding regions are close to Gaussian masks.

Visualization of the Segmentation Results We provide more qualitative comparisons in Figure 2 and Figure 3.

[1] This is also used in Section 5.3 to calculate the region entropies.

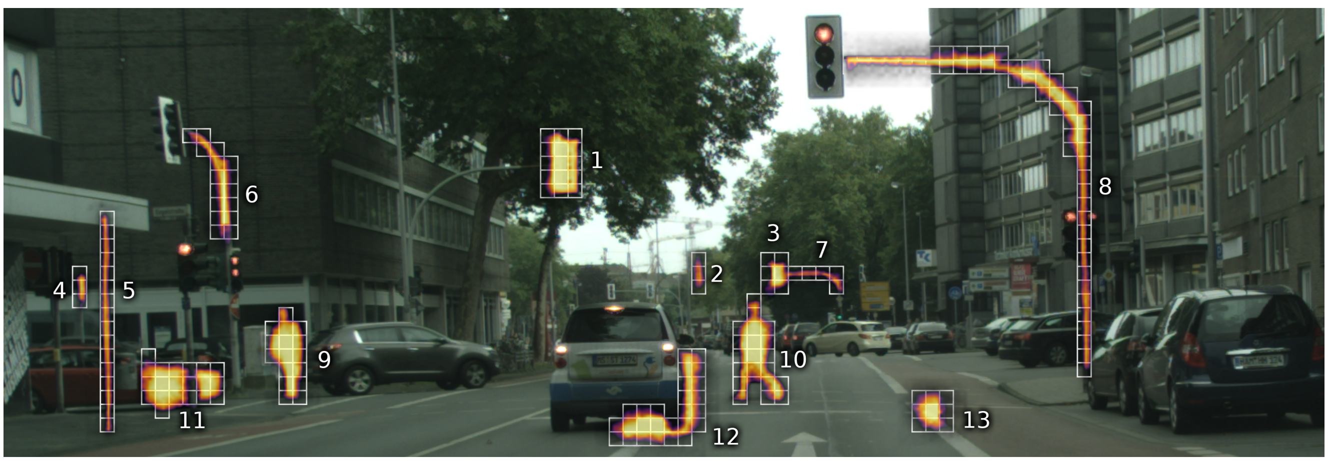

Figure 1. **Geometrics of the leaned *class-agnostic regions*** and its corresponding tokens (marked using white cell) on a Cityscapes validation image. We identify the *token class* for better interpretation: 1~3: `traffic light`; 4: `traffic sign`; 5~8: `pole`; 9~10: `person`; 11: part of `car`; 12: surrounding `road` tokens of a car; 13: inner regions of `road`.

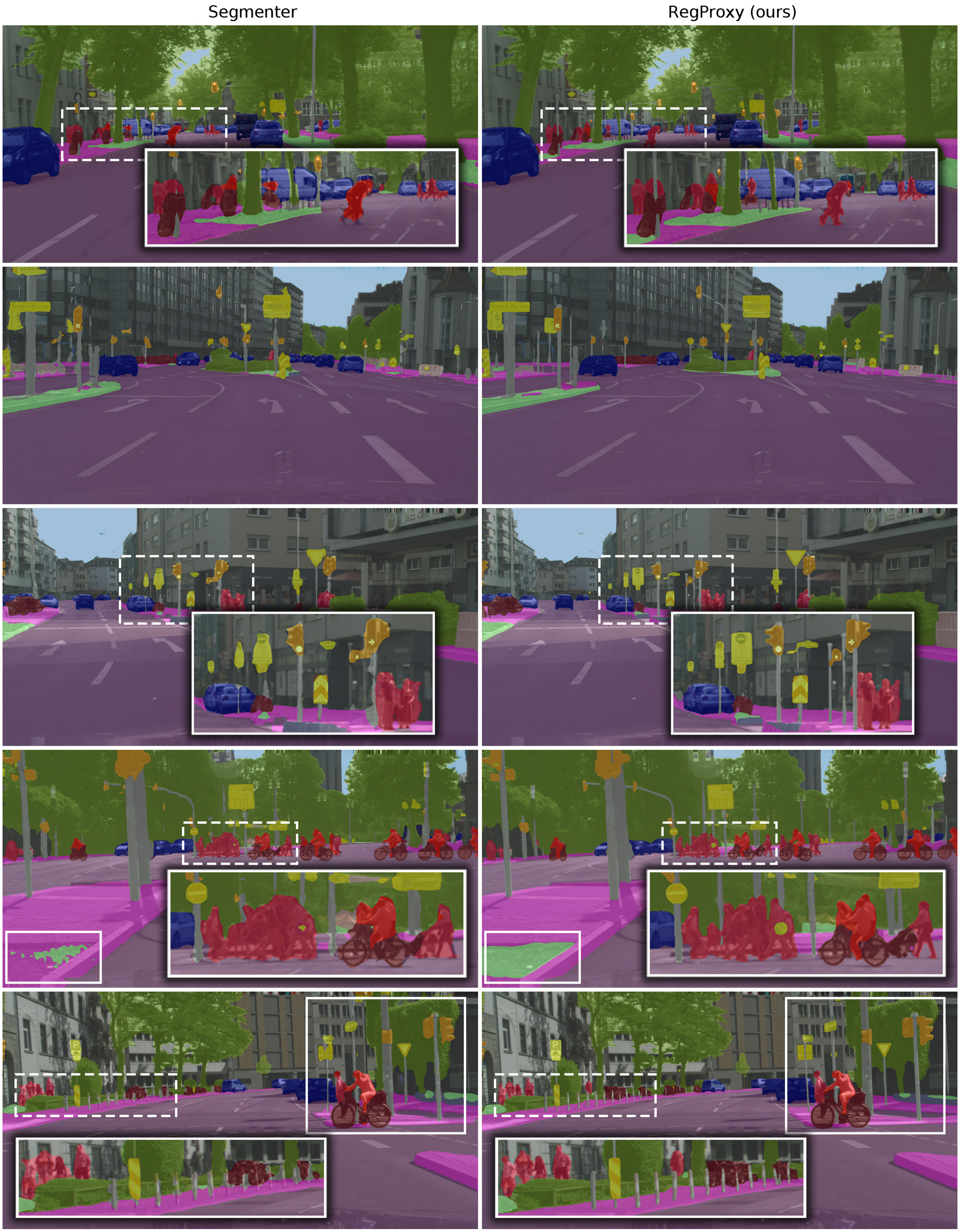

Figure 2. **Qualitative comparison on Cityscapes.**

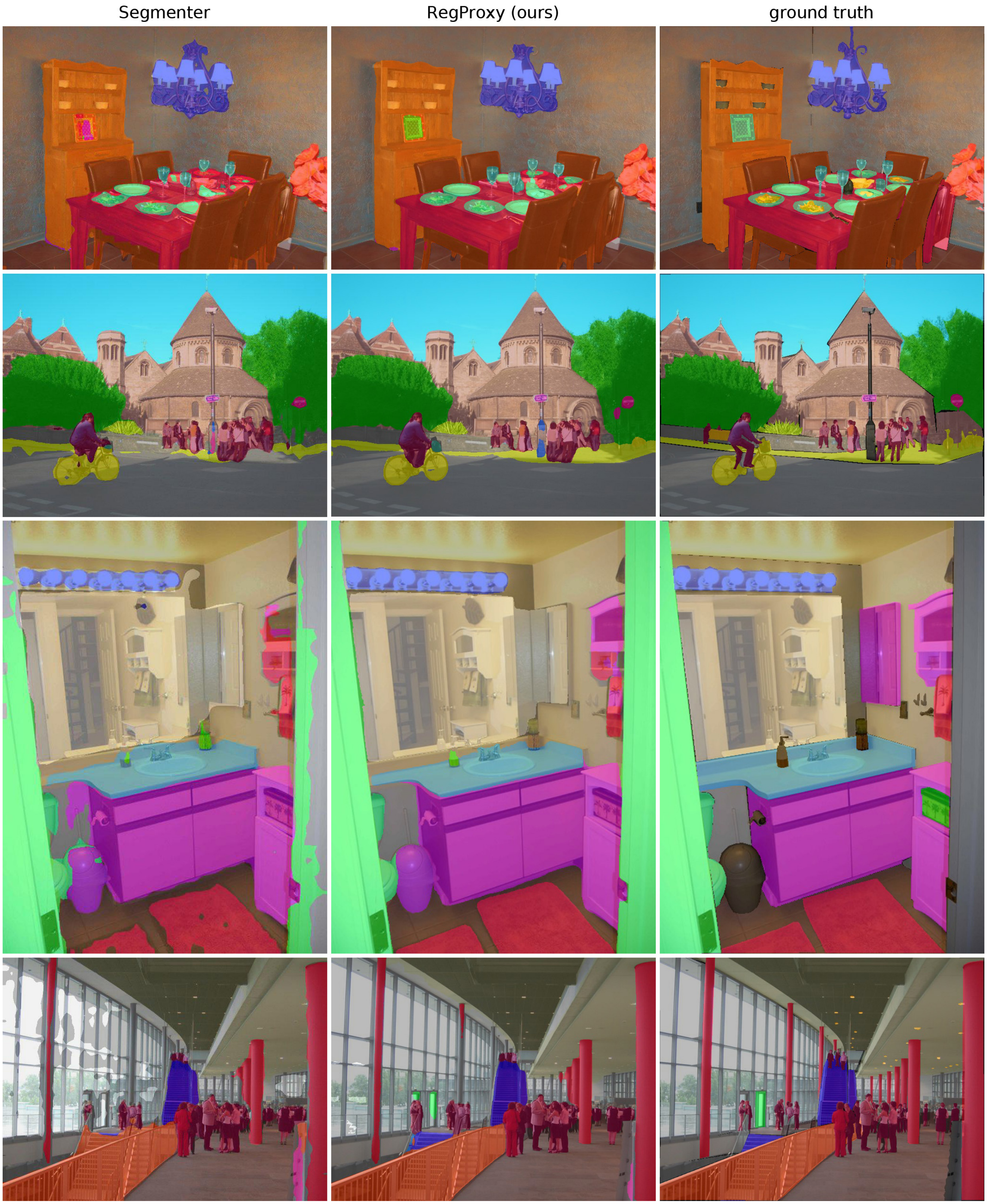

Figure 3. **Qualitative comparison on ADE20K.**

References

[1] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *arXiv preprint arXiv:2104.14294*, 2021. 1, 2

[2] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018. 1

[3] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017. 2

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1

[5] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021. 1

[6] Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your vit? data, augmentation, and regularization in vision transformers. *arXiv preprint arXiv:2106.10270*, 2021. 1, 2

[7] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. *arXiv preprint arXiv:2105.05633*, 2021. 1, 2

[8] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021. 1, 2

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 1

[10] Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-contextual representations for semantic segmentation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16*, pages 173–190. Springer, 2020. 1

[11] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6881–6890, 2021. 1