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 $qkv$ self-attention proposed in [9]. The query, key and value $q, k, v \in \mathbb{R}^{N \times D_h}$ are linearly projected from the input tokens $X \in \mathbb{R}^{N \times D}$. Each output token is a weighted sum over all values $v$ in the sequence, where the weights $A_{ij}$ are based on the pairwise similarity between two elements in the sequence with respect to their $q_i, k_j$ representations. Self-attention can be formulated as:

$$\begin{align*}
[q, k, v] &= \mathbf{X} U_{qkv}, \\
U_{qkv} &\in \mathbb{R}^{D \times 3D_h}, \\
A &= \text{softmax}\left(\frac{qk^T}{\sqrt{D_h}}\right), \\
SA(X) &= Av.
\end{align*}$$

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:

$$\text{MSA}(X) = [SA_1(X), SA_2(X), \ldots, SA_k(X)] U_{msa},$$

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

B. More Implementation Details

**Algorithm 1** Code of “painting” in a PyTorch-like style.

```python
# 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 (ih) (ow) 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 $Q$.

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.

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

<table>
<thead>
<tr>
<th>arch.</th>
<th>$h \times w$</th>
<th>stride</th>
<th>GFLOPs</th>
<th>#params</th>
<th>ADE20K</th>
<th>City.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViT-T/16</td>
<td>- -</td>
<td>16</td>
<td>3.8G</td>
<td>5.7M</td>
<td>38.76</td>
<td>72.02</td>
</tr>
<tr>
<td>RegProxy-T/16</td>
<td>4 x 4</td>
<td>4</td>
<td>40.76</td>
<td>75.08</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RegProxy-S/16</td>
<td>4 x 4</td>
<td>8</td>
<td>40.76</td>
<td>75.08</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RegProxy-S/16</td>
<td>16 x 16</td>
<td>16</td>
<td>41.01</td>
<td>75.15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ViT-S/16</td>
<td>- -</td>
<td>16</td>
<td>14.9G</td>
<td>22.0M</td>
<td>45.04</td>
<td>75.39</td>
</tr>
<tr>
<td>RegProxy-S/16</td>
<td>4 x 4</td>
<td>8</td>
<td>41.18</td>
<td>75.18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RegProxy-S/16</td>
<td>16 x 16</td>
<td>16</td>
<td>47.20</td>
<td>78.37</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 $3 \times 3$ Conv The $3 \times 3$ 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 $8$) yields similar results with the depth-wise one. To sum up, the $3 \times 3$ 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...
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).

Table 2. Effect of the $3 \times 3$ conv.

Effect of explicit regularization on region semantics.

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 $^1$:

$$\text{hist}(s) = L_2 \left( \sum_{p \in N_p} q_r(p) \cdot \text{onehot}(\hat{y}(p)) \right) ,$$

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

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.

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.
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


