```
# x_t[B, L]       - minibatch of texts
# x_v[B, 3, H, W] - minibatch of images
# text_enc        - encode text (x_t) to text emb (t)
# image_enc       - encode image (x_v) to global image emb (v_g) and dense feature (v_d)
# grounding_dec   - decode dense feature (v_d) to pixel-level dense image emb (v_s)
# proj            - scalar projection and sigmoid layer
# gumbel_max      - Gumbel-Max function
# ld_area         - weight of area TCL loss

t = text_enc(x_t)  # t[B, C]
_, v_d = image_enc(x_v)  # v_d[B, L, C]
v_s = grounding_dec(v_d)  # v_s[B, C, H, W]
mask = proj(einsum("ichw,jc->ijhw", v_s, t))  # [B, B, H, W]

pos_mask = mask[arange(B), arange(B)].unsqueeze(1)  # [B, 1, H, W]
pos_mask_b = gumbel_max(pos_mask)  # binarized positive mask
v_g, _ = image_enc(pos_mask_b * x_v)  # v_g[B, C]
s = v_g @ t.T  # s[B, B]
tcl_v = info_nce(s)  # InfoNCE loss

w = mask / mask.sum((2, 3), keepdim=True)
grounded_emb = einsum("ichw,ijhw->ijc", v_s, w)
s = einsum("ijc,jc->ij", grounded_emb, t)  # s[B, B]
tcl_f = info_nce(s)  # InfoNCE loss

pos_area = pos_mask.mean()
neg_area = off_diag(mask).mean()  # average of off-diagonal
tcl_area = ld_area * ((pos_area - 0.4).abs() + neg_area)

tcl_loss = tcl_v + tcl_f + tcl_area
```

Figure 6. **PyTorch-like pseudo code for the core implementation of TCL.**

A. Pseudo-code

For clarity, we present the pseudo-code for the core implementation of TCL in Fig. 6. As described in Sec. 3, we first generate text-grounded masks via grounder and use them to compute text-grounded images and embeddings (L10-L17). Furthermore, this pseudo-code also demonstrates our efficiency-aware design of TCL. We compute the $B \times B$ mask, where B is the batch size, using a single `einsum` operation and scalar projection (L13). When computing TCL_v loss, we need to perform CLIP image encoder inference again for the grounded images (L17). To reduce computational complexity, we only use the positive mask, which is the diagonal of the quadratic mask, for linear inference with a size of B instead of quadratic (L15). When computing TCL_f loss, we use the entire quadratic mask because a single `einsum` operation can efficiently compute the grounded embeddings without requiring additional encoder inference (L22). A more detailed discussion on the efficiency is provided in the later section (Appendix D).

B. Architecture Details

Our core design principle is to preserve and exploit the diverse knowledge of pre-trained CLIP [23]. Therefore, we freeze the pre-trained CLIP encoders[2] and train the grounding decoder for the adaptation from the image-text alignment to the region-text alignment. We also considered the other techniques to preserve knowledge [4], but a simple freezing strategy worked the best. We follow the simple modification of MaskCLIP [33] to the CLIP image encoder.

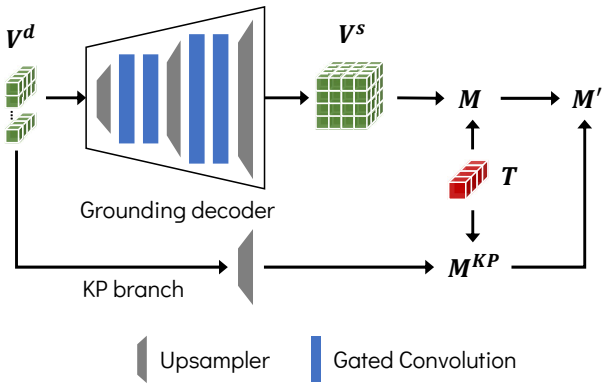


Figure 7. **Architecture of the grounding decoder.** The knowledge preservation (KP) branch serves to preserve pre-trained knowledge intact.

[2] After 30,000 iterations, we unfreeze only the last block of the image encoder for richer model capability.

Methods	PAMR	with background class			without background class					Avg.
		VOC	Context	Object	VOC20	Context59	Stuff	City	ADE	
GroupViT		50.4	18.7	27.5	**79.7**	23.4	15.3	11.1	9.2	29.4
MaskCLIP		38.8	23.6	20.6	74.9	26.4	16.4	12.6	9.8	27.9
ReCo		25.1	19.9	15.7	57.7	22.3	14.8	21.1	11.2	23.5
TCL (Ours)		**51.2**	**24.3**	**30.4**	77.5	**30.3**	**19.6**	**23.1**	**14.9**	**33.9**
GroupViT	✔	51.1	19.0	27.9	81.5	23.8	15.4	11.6	9.4	30.0
MaskCLIP	✔	37.2	22.6	18.9	72.1	25.3	15.1	11.2	9.0	26.4
ReCo	✔	27.2	21.9	17.3	62.4	24.7	16.3	22.8	12.4	25.6
TCL (Ours)	✔	**55.0**	**30.4**	**31.6**	**83.2**	**33.9**	**22.4**	**24.0**	**17.1**	**37.2**

Table 3. **Standardizing PAMR condition.** To reveal the effect of our refinement method (PAMR), we compare the methods with the same PAMR condition. TCL achieves the best in both with and without PAMR settings.

They modify the last attention of the CLIP image encoder to acquire the dense features representing local semantics. These dense image features $\mathbf{V}^d$ are fed to the grounding decoder. As shown in Fig. 7, the grounding decoder consists of four gated convolution blocks, where the output of a convolution is gated by a learned gating parameter and added to the skip connection. Concretely, the process of gated convolution can be written as:

$$\mathbf{x}' = \mathbf{x} + \tanh(g) \cdot \text{Conv}(\mathbf{x}), \tag{14}$$

where $\mathbf{x}$ is input feature and g is a learned gating parameter. The upsamplers increase the feature map resolution for the high-resolution segmentation capability. The first two upsamplers use the nearest neighbor interpolation and the last upsampler adapts the resulting embedding into the pixel-level embedding by the bilinear interpolation. In addition, as shown in Fig. 7, we employ two branches strategy: the main grounding decoder and knowledge preservation (KP) branches. In this KP branch, the CLIP dense features $\mathbf{V}^d$ are reshaped spatially and upsampled to pixel-level resolution by bilinear interpolation, and then we compute the text-grounded mask $\mathbf{M}^{\text{KP}}$ by Eq. (5). There are no learnable parameters in this branch and the output masks are just mixed with the output from the grounding decoder branch as follows:

$$\mathbf{M}' = (1 - w_{kp}) \cdot \mathbf{M} + w_{kp} \cdot \mathbf{M}^{\text{KP}}, \tag{15}$$

where $\mathbf{M}^{\text{KP}}$ is the generated masks from knowledge preservation branch, $\mathbf{M}'$ is the final output mask, and w_{kp} is a mixing hyperparameter. We use the w_{kp} of 0.3. To fully leverage the massive pre-trained knowledge of CLIP [23], this branch is only used in the inference stage. It also can be regarded as a cost-free ensemble.

C. Fair Comparison

In this paper, we present a unified evaluation protocol to facilitate a fair and rigorous comparison. However, the condition of fair evaluation protocol can be controversial. Thus, in this section, we provide additional comparisons to enhance fairness, paving the way for future research on fair comparison in open-world segmentation.

Fair comparison with refinement methods. In our unified evaluation protocol, we do not unify refinement methods, as we believe that each method's approach to refining the model output is a design choice. However, some may argue that a fair comparison protocol should unify the refinement methods as well. To address this concern, we also provide the comparison with and without PAMR [1], which is our refinement method used in TCL. As shown in Table 3, even using the same refinement method across all methods, TCL still achieves state-of-the-art performance with a significant margin in both settings, demonstrating the effectiveness of its underlying approach. It is also worth noting that the performance gains resulting from PAMR are specific to each method. For example, TCL and ReCo demonstrate significant performance improvements of $+3.3$ and $+2.1$ mIoU, respectively, while GroupViT only shows a marginal gain of $+0.6$ mIoU, and MaskCLIP actually leads to a decrease in performance of -1.5 mIoU. We believe that this is because each comparison method was designed without considering refinement by PAMR. Thus, because the effectiveness of the refinement method is closely tied to the main method, we have not unified the use of refinement methods in our proposed evaluation protocol.

Fair comparison in dataset scale. Except for GroupViT, all comparison methods (TCL, ReCo, and MaskCLIP) utilize CLIP pre-trained models. GroupViT proposes a new encoder architecture and is therefore unable to leverage CLIP pre-trained models directly, which is one of its limitations. Hence, a fair comparison would be to evaluate GroupViT as it is. Nevertheless, it could be argued that comparing GroupViT and other methods at the same training dataset scale would be a fair comparison. To address this concern, we provide additional scale-up experiments. As mentioned, GroupViT is unable to leverage CLIP pre-

Methods	Datasets	VOC20	Context59	Avg.
GroupViT	CC15M + RedCaps12M	79.7	23.4	51.6
	CC15M + YFCC14M	74.1	20.8	47.5
	CC15M + Coyo100M	73.3	25.0	49.2
	CC15M + Coyo700M	75.5	24.2	49.9
TCL (Ours)	CC15M + WIT400M	**83.2**	**33.9**	**58.6**

Table 4. **Scale-up GroupViT** does not directly help the segmentation capability. WIT400M indicates the dataset of CLIP pre-training.

	TCL	TCL†	GroupViT	MaskCLIP	ReCo
Speed (s)	0.08	0.07	0.05	0.04	28.10
FPS (it/s)	12.93	15.11	20.94	26.09	0.04
Mask ratio	$1/4^2$	$1/4^2$	$1/16^2$	$1/16^2$	$1/16^2$

Table 5. **Inference speeds and FPS.** Mask ratio indicates the resolution ratio of the segmentation masks compared to the input image size. For example, TCL generates 112×112 masks for 448×448 input image, while GroupViT generates 28×28 masks. TCL† denotes the TCL model without PAMR.

trained models directly. Thus, for a fair comparison, we train GroupViT using the publicly available Coyo700M dataset of large-scale image-text pairs [2], which is larger than the non-public CLIP training dataset (WIT400M). However, we observe that a simple scale-up of the training dataset does not guarantee improvement in performance. As shown in Table 4, the correlation between dataset size and performance is not clear. The impact of larger datasets varies between the datasets (*e.g.*, "CC15M+RedCaps12M" performs best on VOC20, but "CC15M+Coyo100M" performs best on Context59), and TCL outperforms all variants of GroupViT despite the fair dataset scale.

D. Efficiency Analysis

While efficiency is not the primary objective of this study, it is also considered one of our core design principles, especially regarding inference efficiency for practical applications. This section provides an analysis of the inference throughput and our design choices for efficiency.

Inference throughput. We benchmark the inference speeds and FPS using 448×448 images and 21 target classes on a single NVIDIA V100 GPU. Our benchmark setting follows an open-world scenario that addresses an arbitrary class, meaning that the text embeddings are computed for every inference. As shown in Table 5, TCL has slightly lower FPS compared to GroupViT or MaskCLIP, due to the relatively high resolution of segmentation masks. This trade-off between mask resolution and FPS can be controlled by the design of the grounding decoder, but we do not investigate this further, as it is outside the scope of this study. ReCo [28] shows very low FPS due to its retrieval process, which involves retrieving similar images from the ImageNet dataset [25] and co-segmenting them. Note that there are many methods to reduce the computational cost of TCL, *e.g.*, utilizing depthwise convolution instead of standard convolution [7], but it is not the focus of this study and is left for future work.

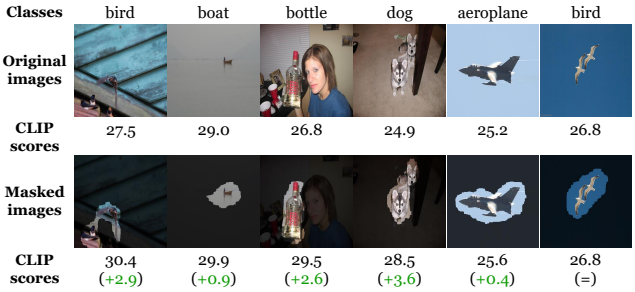


Figure 8. **Qualitative examples on the robustness of CLIP for masked images.** The first and second rows show natural (non-masked) and masked images, respectively. The number below each image indicates CLIP score between the image and the corresponding class.

Efficiency-aware designs of TCL. In contrastive learning, the image-text alignment requires $B \times B$ matching, where B denotes either the batch size in training or the number of target classes in testing. This quadratic operation is a major bottleneck when scaling up the batch size. To address this, CLIP [23] uses a late fusion architecture to minimize the $B \times B$ operation. In this architecture, the encoders are decoupled, and the encoded embeddings are combined at the end of the network, making the model efficient in both training and inference. We also employ the late fusion architecture following CLIP for efficiency. In addition, since TCL$_v$ requires additional encoding of text-grounded images ($\tilde{\mathbf{x}}^V$), we use $\tilde{\mathbf{x}}^V$ from only positive pairs (=B encodings), and bypass $B \times B$ encodings by introducing TCL$_f$ to compute $\mathbf{v}^f$ without any additional encoding. We further improve efficiency by freezing encoders and reusing text embeddings. As a result, our entire training time is about 12 hours, and our inference throughput is comparable to MaskCLIP or GroupViT. Nevertheless, it is also worth noting that scaling up TCL is more challenging than CLIP since the $B \times B$ mask is a spatial tensor ($B \times B \times H \times W$), unlike CLIP. Empirically, we found that algorithmic improvements on the scale should be considered to scale up to $B > 8,192$.

E. Additional Details and Experiments

CLIP with masked images. TCL implicitly assumes that CLIP can address masked images robustly. While CLIP is widely known for its strong robustness [23], it is not yet

Methods	VOC20	Context59	Stuff	Avg.
ViL-Seg	34.4	16.3	16.4	22.4
ReCo	57.9	32.0	18.4	36.1
GroupViT (RedCaps)	79.0	49.2	16.1	48.1
MaskCLIP	73.0	56.5	21.9	50.5
TCL (Ours)	**84.5**	**62.0**	**27.6**	**58.0**

Table 6. **Zero-shot segmentation performance for partial classes.** The numbers of ViL-Seg are adopted from the original paper [19]. We use the proposed unified evaluation protocol for the other results.

clear how well CLIP can handle masked images. As such, we investigate CLIP scores between masked images and positive texts in Fig. 8. The results indicate that CLIP can address masked images. Interestingly, the scores tend to be improved, especially for the images with complex context (columns 1, 3, and 4 of Fig. 8). The masks remove noisy context and help to focus on the target object, leading to improved scores.

Details on the comparison methods. In the quantitative evaluation, we include the variants of the comparison baselines for an extensive comparison; the GroupViT variants from the YFCC and RedCaps checkpoints [30] and the MaskCLIP variants by the refinement process (key smoothing and prompt denoising). For the backbone of MaskCLIP, we use ViT-B/16 since its reported performance is better than ResNet-50 [33]. For the qualitative comparison, we choose the quantitatively best variant for each method.

Comparisons with zero-shot semantic segmentation methods. We provide an extensive and unified comparison in Table 1, but the comparison does not include non-open-sourced methods. To the best of our knowledge, ViL-Seg [19] is the only non-open-sourced method for open-world semantic segmentation. We compare the zero-shot segmentation performance following the evaluation protocol of ViL-Seg. In this evaluation protocol for zero-shot semantic segmentation, only partial classes are used: 5 classes (potted plant, sheep, sofa, train, tv-monitor) for PASCAL VOC20, 4 classes (cow, motorbike, sofa, cat) for PASCAL Context59, and 15 classes (frisbee, skateboard, cardboard, carrot, scissors, suitcase, giraffe, cow, road, wall concrete, tree, grass, river, clouds, playingfield) for the COCO-Stuff dataset. Therefore, we additionally provide the comparison results under the partial classes protocol. As shown in Table 6, TCL achieves state-of-the-art performance with a large margin in every dataset again.

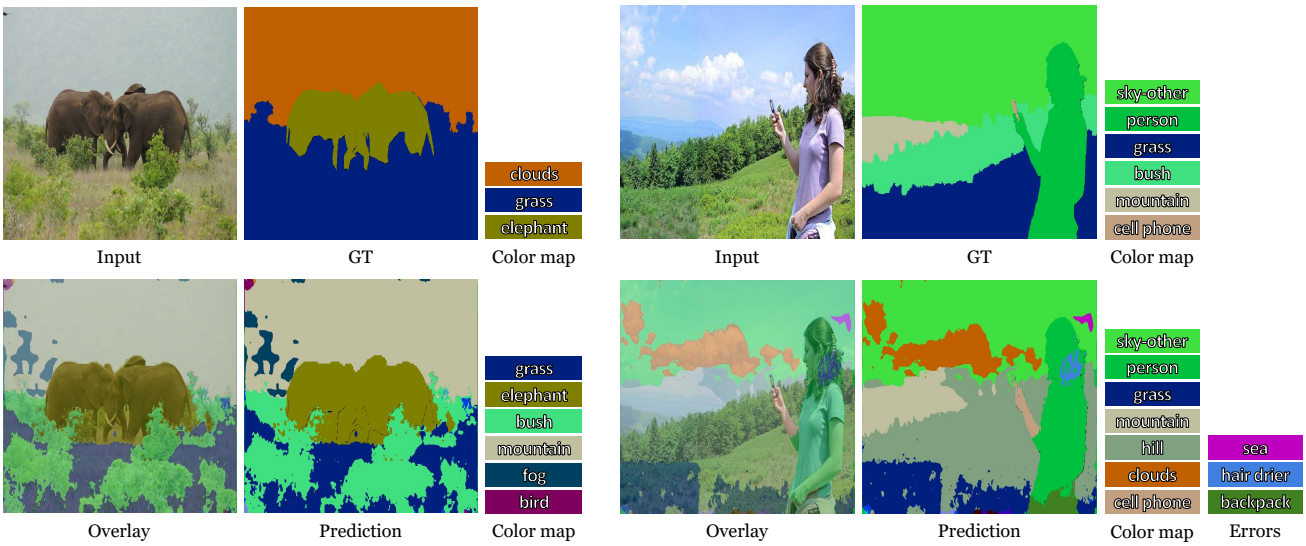
F. Case Study on Failures

Errors of TCL. We investigate the failure cases of TCL via various qualitative examples. First, TCL undergoes difficulty in capturing segment boundaries accurately. For example, in Fig. 9b, the predicted segment of the "mountain" class includes part of the sky region, and the "cell phone" segment contains the right hand and arm regions. This is a fundamental challenge of unsupervised open-world segmentation; the absence of dense annotation makes precisely capturing a segment boundary extremely difficult. Although the proposed method remarkably improves the segmentation performance compared with previous methods, this case study reveals that there are still many areas to be improved. Furthermore, despite the help of the smooth prior loss, the predictions still tend to be noisy, *e.g.*, "sea" or "hair drier" in Fig. 9b.

Ambiguity in benchmarks. On the other hand, we also find crucial issues in the current benchmark datasets: ambiguities in the class label set and scene semantics. In particular, there are lots of class labels with similar semantics, especially in the datasets with a large vocabulary, *e.g.*, COCO-Stuff (171 classes) [3] or ADE20K (150 classes) [32]. Mostly the labels have different semantics in detail, but the distinction between the labels can be ambiguous depending on how the image captures the scene. For example, it is hard to distinguish "clouds" and "fog" in Fig. 9a and "hill" and "mountain" in Fig. 9b. Also, there are labels with superset-subset relations. For instance, the COCO-Stuff dataset has "broccoli", "vegetable", and "food-other" classes. In the supervised setting, a model can address this issue by training only if there is labeling consistency between images. However, in the open-world scenario, such superset-subset relations cause significant ambiguity. Furthermore, a more frequent ambiguity raises when a segment has multiple semantics. More proper descriptions of the "clouds" and "grass" segments in Fig. 9a are "foggy or cloudy mountain" and "bushes on the grass", respectively. However, ground truth (GT) labels represent only part of the entire semantics. As with the superset-subset relation case, benchmarks for the open-world scenario require additional consideration to address such ambiguities. In this study, we propose a unified evaluation protocol to compare the existing methods fairly, but it only unifies the evaluation protocol and simply employs existing benchmark datasets. This analysis suggests the need for further advanced benchmarks dedicated to open-world scenarios in the future.

G. Analysis on Model Behavior

In this section, we investigate how the learned TCL model generates different segmentation masks depending on the input text prompts. As shown in Fig. 10, the model tends to capture the intended region better when the input text prompt is more specific. Although this characteristic can cause performance degradation when evaluating the model on a fixed benchmark, it also improves the



(a) Case 1 (b) Case 2

Figure 9. **Ambiguous error cases.** The color map of a negligible region is omitted. Although different from the ground truth (GT), there are many cases that can be considered correct. The segments in the color map are semantically correct despite of difference with GT, while the errors indicate clearly incorrect predictions.

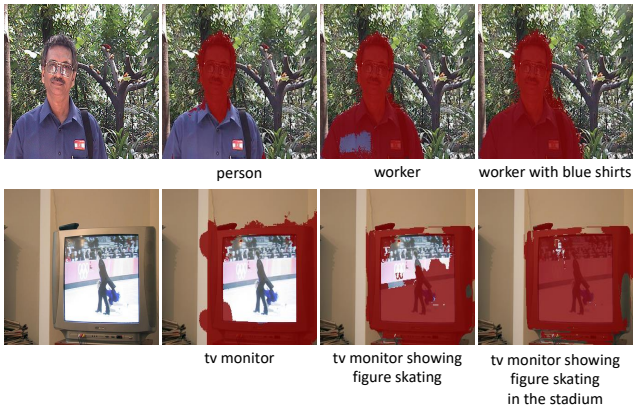


Figure 10. **TCL behaviors depending on the specificity of the prompts.** The red-colored region indicates the segmentation results of the given text prompts. The more specific the prompt, the better the segmentation result.

controllability of the model. We can exploit this controllability to maximize the benchmark performance, *e.g.*, class name expansion. However, we do not employ these dataset-dependent tricks to prevent the overestimation of the model performance, as described in Sec. 4.1.

H. Additional Qualitative Results

H.1. Qualitative Examples on Complicated Scene

Qualitative comparison on PASCAL VOC in Sec. 4.3 visualizes the performance difference between comparison methods. However, the VOC dataset tends to be object-oriented and its images are generally composed of one or two segments. In this section, we qualitatively compare open-world segmentation methods in complicated scenes. Figs. 11 and 12 show examples including at least 3 segments from the Cityscapes and COCO-Stuff datasets. As shown in the figures, GroupViT [30] and MaskCLIP [33] tend to generate a small number of segments. It makes the results less noisy but causes a large error. For example, in Fig. 11, MaskCLIP fails to segment "building" regions and GroupViT misidentifies "road" as "traffic light". In contrast, ReCo [28] suffers from noisy prediction. Our TCL also generates partially noisy results, but it is relatively cleaner and better than the comparison baselines. It is also worth noting that the image resolution of the unified evaluation protocol is relatively smaller than the widely used protocols for Cityscapes. We resize a shorter side of an image to 448 with keeping the aspect ratio, resulting in 448×896. In contrast, 1024×2048 is the widely used resolution for the Cityscapes dataset [6, 29]. Increasing the resolution can help recognize small objects, *e.g.*, the persons in Fig. 11.

H.2. Additional Qualitative Examples in the Wild

Fig. 13 shows additional qualitative examples from web images in the wild. In this experiment, we investigate the discrimination capability of the model in various aspects: proper nouns (Frodo, Gandalf, Pyramid, Sphinx, Samwise, Gollum, Taj Mahal, Batman, Superman), colors with the same object (red, green, yellow bananas), letters (MMU, Turkish, Fighter), and subclasses (Corgi, Shepherd). The results show that our model can recognize and segment var-

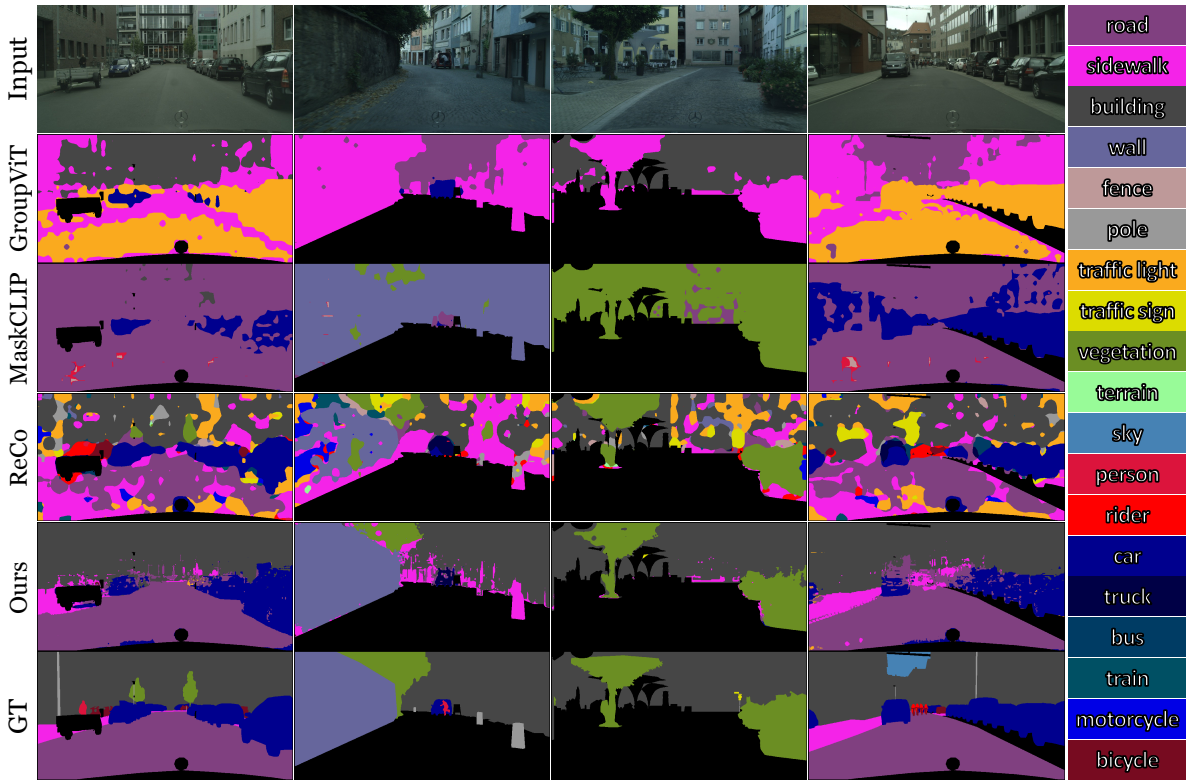


Figure 11. **Additional qualitative examples on Cityscapes.**

ious concepts in the wild. For the baseline models, the results show similar tendencies with the in-the-wild examples in Sec. 4.3. Contrary to the quantitative evaluation in fixed benchmarks, ReCo [28] generates a relatively plausible segmentation map compared with GroupViT [30] and MaskCLIP [33].

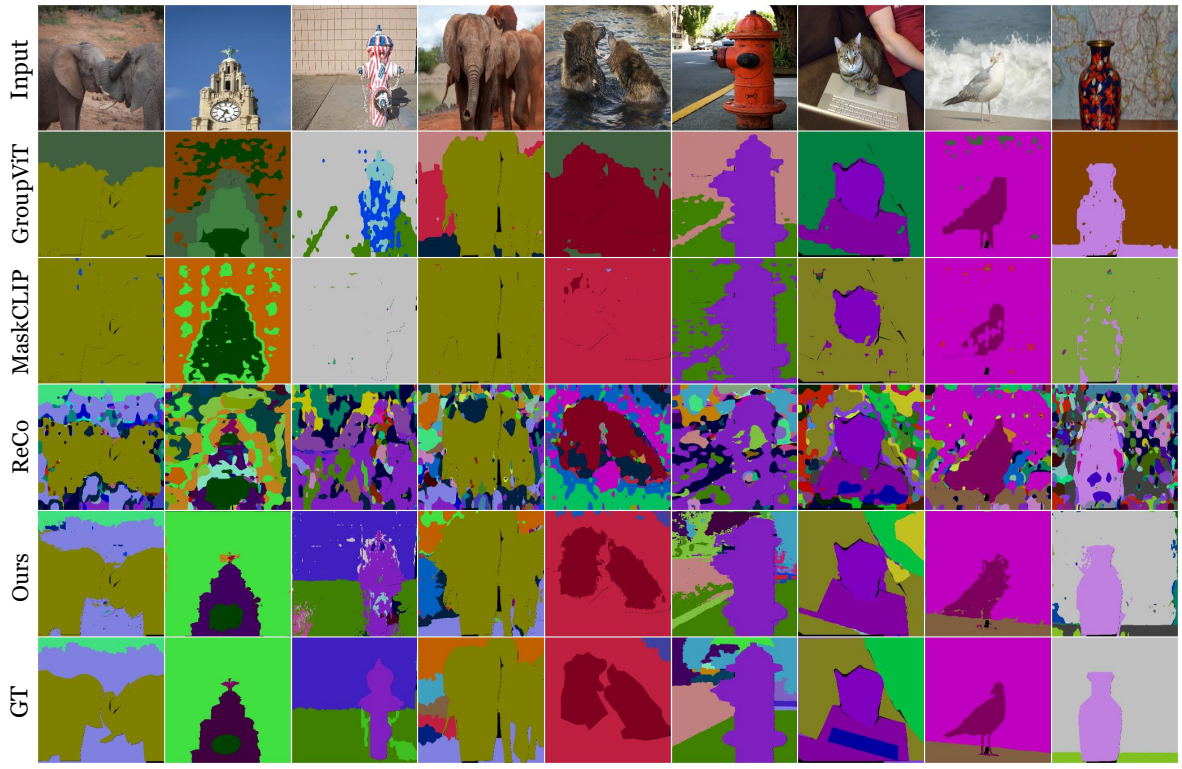


Figure 12. **Additional qualitative examples on COCO-Stuff.** Color map is omitted since COCO-Stuff has 171 classes.

Figure 13. **Additional qualitative examples in the wild.**