A Potential societal impacts

OpenSeg provides a language interface for recognition and localization of visual concepts. Such an interface may facilitate downstream applications such as interactive intelligent home assistants, interactive content creation, and instructing robot actions with language [43]. However, OpenSeg models are trained on large-scale datasets, which may contain bias towards certain image and text distributions. As a result, we share the similar concern as recent studies for evaluating image-text models like Clip [1]. Thus OpenSeg is not suitable for deployment in the real world without properly studying the model biases and calibrating the predictions.

B Limitations of our approach

The limitations come with the strength of our method. The mask representations can organize an image into a small number of regions, and thus enable scalable visual-semantic alignments. However, our method will not be able to segment visual concepts that do not have associated segmentation proposals. That being said, we still believe the generalization to unseen concepts with class-agnostic mask representations is easier than pixel-wise representations. How to improve generalization or adaptation of mask predictions will be an interesting research topic.

C Architecture of the cross-attention module

Figure 6 shows the model architecture of the cross-attention module. The mask queries $\mathbf{q}^{(t)}$ interact with the position-augmented image features $\mathcal{F}_f + PE$ to generate mask queries $\mathbf{q}^{(t+1)}$. The first mask queries $\mathbf{q}^{(0)}$ is randomly initialized at the beginning of training. The module is stacked three times ($T = 3$) in our experiments. We try to include self-attention of queries followed by the query-image cross-attention as in [10], but that does not improve performance.

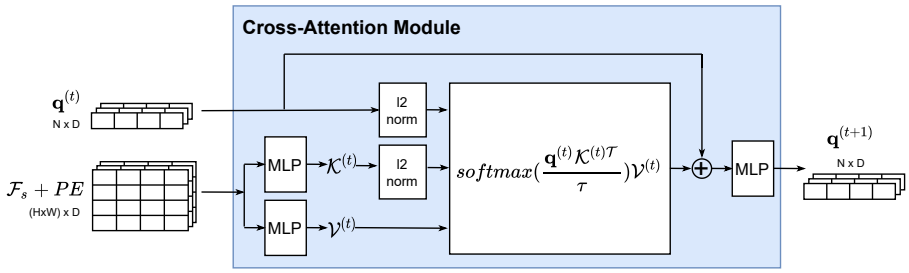

Figure 6. Model architecture of the region-image cross-attention module. The module is stacked repeatedly to generate mask queries for region segmentation.

Table 6. Our mask prediction can generalize across datasets. We report the recall at IoU 0.5.

Train / Test	COCO	ADE20K	Mapillary	IDD	BDD	Cityscapes	SUN
Cityscapes	28.8	25.8	42.5	50.0	61.2	69.3	22.4
COCO	82.2	68.0	48.6	58.1	64.2	58.6	90.8
MSeg	82.9	80.3	55.8	68.1	73.4	67.2	93.6

D Mask generalization on MSeg dataset

In this experiments, we use the setup and the curated annotations in MSeg [28]. Our goal is to verify if a model trained on a single dataset can generalize to multiple datasets. Table 6 summarizes the results of recall at an IoU of 0.5. The model achieves the best results when trained on MSeg, which aggregates the training images and annotations of all datasets, and can be seen as the performance upper bound. The results are slightly worse when trained on COCO, showing the model trained on COCO can generalize reasonably well. The model trained on Cityscapes generalizes poorly, indicating OpenSeg requires a large and diverse training dataset to provide class-agnostic segmentation proposals.

E Visualization of full mask proposals

In Figure 4 of the main paper we present a subset of predicted segmentation masks in an unseen scene. Figure 7 shows all the 128 mask proposals on the same image.

F Ablation on batch size

In all of the experiments unless otherwise stated, we use a global batch size of 1024 and our local batch size is 16 (we have 64 cores). Also, we compute the contrastive loss over the local examples of each core. Therefore, the effective batch size of loss is equal to local batch size. In this section, we compare OpenSeg with unsync and sync contrastive loss. In the sync version, we compute the loss over all cores and effective batch size of loss is equal to global batch size. As shown in Table 7, sync loss improves the performance of OpenSeg. We also train OpenSeg with different batch sizes (same epochs). We find that OpenSeg is robust to the batch size.

G Importance of segmentation loss

Since grounding loss $\mathcal{L}_{\mathcal{G}}$ back-propagates through the segmentation head, it is possible to train the complete OpenSeg model with only the grounding loss. In this section we study the importance of having segmentation loss in addition to the grounding loss in training OpenSeg. In Table 8, we compare the performance

Table 7. OpenSeg is robust to the batch size. We present performance of OpenSeg trained on COCO+Loc. Narr. and different batch sizes. Numbers inside the parentheses represent effective batch size for the contrastive loss.

batch size	steps	A-847	PC-459	A-150	PC-59
1024 (16)	60k	6.8	11.2	24.8	45.9
256 (256)	240k	8.1	11.2	26.2	45.4
512 (512)	120k	8.2	11.2	25.7	45.0
1024 (1024)	60k	8.1	11.5	26.4	44.8
2048 (2048)	30k	8.4	11.4	26.9	45.3

of OpenSeg when it is trained with only grounding loss (second row) vs when it is trained with both losses (first row). The former model has significantly worse performance which illustrates the importance of segmentation loss in learning the visual grouping.

When training with COCO + Loc. Narr., we annotate Loc. Narr. dataset with mask pseudo labels (Section 4.3), so that we can compute both grounding and segmentation losses on all training examples. We study the importance of the pseudo labels and the segmentation loss by setting the weight of segmentation loss to zero on examples from Loc. Narr. In Table 8, we compare the performance of these two approaches. The model trained with mask pseudo labels (third row) has worse mIoU on `PC-59`. However, it has better performance on `A-847`, `PC-459` and `A-150` datasets, which include categories outside the COCO dataset. These results indicate that adding mask pseudo labels and computing the segmentation loss on all of the training examples helps the generalization of OpenSeg.

Table 8. Grounding loss is not sufficient for training OpenSeg. We experiment with setting segmentation loss to zero on COCO examples when training on COCO, or on Loc. Narr. examples when training on COCO + Loc. Narr. In both cases the performance of the model drops.

	Segmentation loss							
	COCO	Loc. Narr.	A-847		PC-459		A-150	PC-59
OpenSeg(COCO)	✓	-	6.3		9.0		21.1	42.1
OpenSeg(COCO)	✗	-	(-3.8)	2.5	(-5.7)	3.3	(-14.6) 6.5	(-26.5) 15.6
OpenSeg(COCO + Loc. Narr.)	✓	✓	6.8		11.2		24.8	45.9
OpenSeg(COCO + Loc. Narr.)	✓	✗	(-0.5)	6.3	(-0.8)	10.4	(-2.4) 22.4	(+1.4) 47.3

H Ablation on randomly dropping words

In our experiments, we extract the list of nouns from the captions and then we keep each word by the probability of 0.75. In Table 9 we compare the performance of OpenSeg with different keeping probabilities. We obtain the best results when the keep prob is 0.5 or 0.75, which shows that randomly dropping words within a certain probability range prevents overfitting and improves the performance.

Table 9. Randomly dropping words improves performance of OpenSeg. We extract the list of nouns from captions and keep each noun by a probability of kp. We get the best results with $kp = 0.5/0.75$.

	A-847	PC-459	A-150	PC-59
$kp = 1.0$	5.8	10.4	22.7	44.9
$kp = 0.75$	**6.8**	**11.2**	**24.8**	**45.9**
$kp = 0.5$	**7.0**	10.7	**25.0**	**46.0**
$kp = 0.25$	**6.8**	9.8	22.4	43.5

I Ensembling and prompt engineering

In this section, we study how we can further improve the performance of OpenSeg by prompt engineering and ensembling with the class names provided by testing segmentation datasets.

Ensembling: An object can often be referred with more than one possible description. Some of them exist in the testing dataset, and some do not. For example, image captions usually include one of the descriptions of 'man, woman, boy, girl, *etc.*' when referring to the 'person' category in the testing segmentation datasets. Thus to further improve the performance, we manually assemble a list of synonyms, subcategories and plurals for some of the categories. Here are a few examples:

- person → person, child, girl, boy, woman, man, people, children, girls, boys, women, men
- dog → dog, puppy, dogs, puppies
- cat → cat, kitty, cats, kitties
- grass → grass, grasses, lawn, turf
- bottle → bottle, bottles, water bottle, water bottles

Polysemy: Some class names of the segmentation datasets are polysemous. As a result, a model may make predictions for different meanings of a concept, while the dataset only includes one of the meanings. For example, the class 'fan' in ADE20k refers to a cooling machine, but OpenSeg sometimes labels a crowd of fans (people) on a stage watching a game as 'fan'. To resolve this issue, we add a short context to some of the labels. *e.g.*, we change 'fan' to 'ceiling fan, floor fan'.

Categories have overlap: Another challenge is that some of the classes of a dataset may have overlap. Although the annotators may follow some rules that prevent the overlap. An example of this can be seen in the Figure 5. `A-847` has both 'roof' and 'building' categories, and both of them are correct labels for the 'roof' region in this figure. Another example is the 'clothes' and 'person' categories in the `A-150` and `PC-59` datasets, where a model is penalized for predicting the 'clothes' category on a person. This issue happens more frequently as the vocabulary size gets larger. We don't have a good solution for this issue. However, in addition to the mIoU metric, we calculate the Grounding mIoU metric that has less of this issue.

Table 10 provides the gains that we attain by applying ensembling and prompt engineering. Overall, the improvement is less significant when we scale up training data from COCO to COCO+Loc. Narr. (+2.4 *vs.* +1.8 on average across 4 benchmarks). Moreover, since there is less ambiguity in terms of class names for the Grounding mIoU, the improvement is smaller for this metric in comparison to mIoU (+1.2 vs +1.8 on average across 4 benchmarks when we train on COCO+Loc. Narr.). We will open source our modified list of class names used for this experiment.

Table 10. Ensembling and prompt engineering improves performance of OpenSeg.

		mIoU								Grounding mIoU							
Training data	eng.	A-847		PC-459		A-150		PC-59		A-847		PC-459		A-150		PC-59	
COCO	✗	6.3		9.0		21.1		42.1		21.8		32.1		41.0		57.2	
COCO	✓	+0.5	6.8	+1.0	10.0	+3.7	24.8	+4.3	46.4	+0.6	22.4	+0.6	32.7	+2.5	43.5	+3.7	60.9
COCO+L.Narr.	✗	6.8		11.2		24.8		45.9		25.4		39.0		45.5		61.5	
COCO+L.Narr.	✓	+0.8	7.6	+0.6	11.8	+2.9	27.7	+3.1	49.0	+0.1	25.5	+0.7	39.7	+1.3	46.8	+2.7	64.2

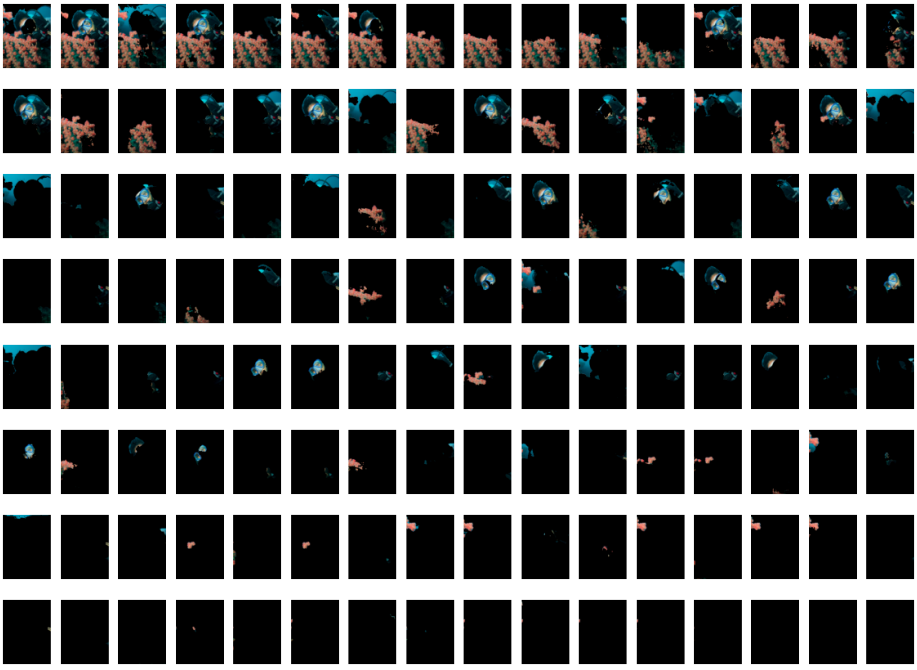

Figure 7. Full set of predicted segmentation masks. This model is trained to predict 128 segmentation masks.

J Visualization of segmentation predictions

In Figures 8-11, we present the predictions of OpenSeg on random images in the `A-150` dataset, where the list of dataset categories are used as the query. For each image, we visualize the output of OpenSeg. We also show the per-pixel prediction without incorporating the mask proposals (see Section 4.4) for comparison.

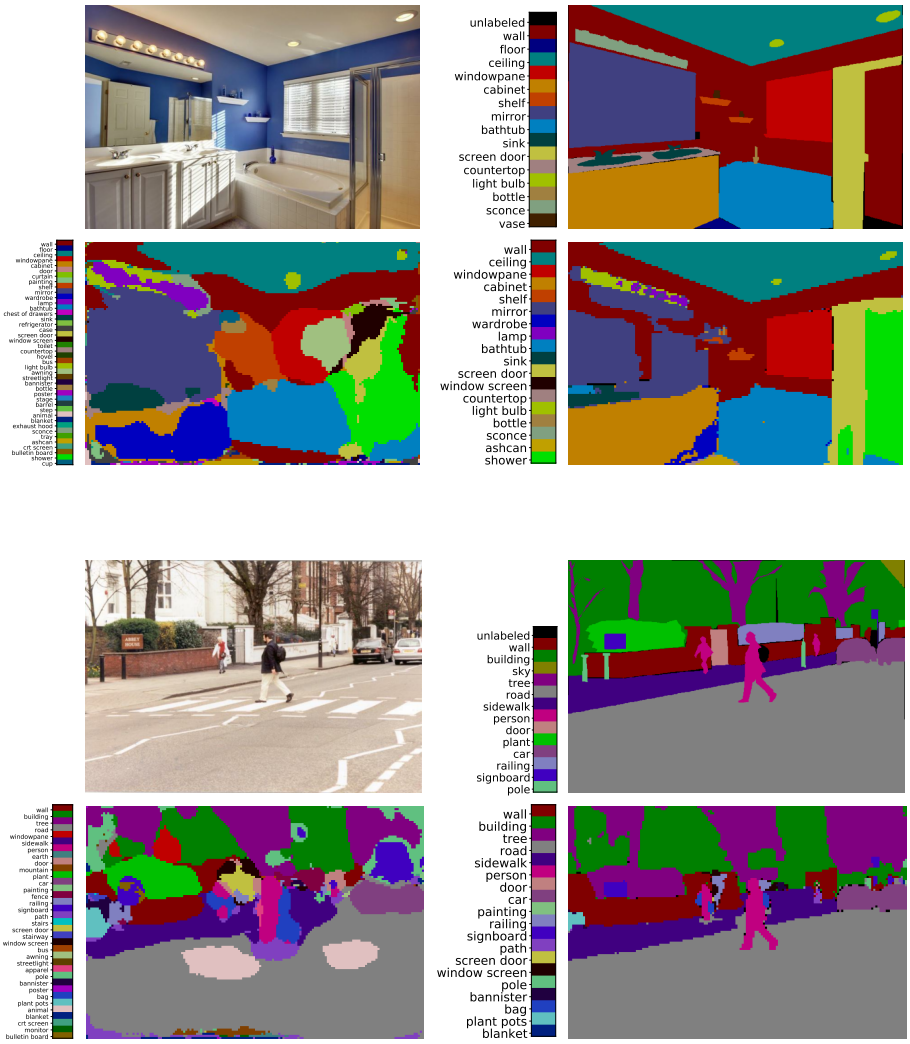

Figure 8. Predictions of OpenSeg on random examples in the `A-150` dataset (Part1). For each example, top left is the input image, top right is the ground-truth mask, and bottom right is the output of OpenSeg. Bottom left shows per-pixel prediction of OpenSeg without incorporating segmentation proposals. Note we only display one name in the legend for each category, but each category may include a list of names.

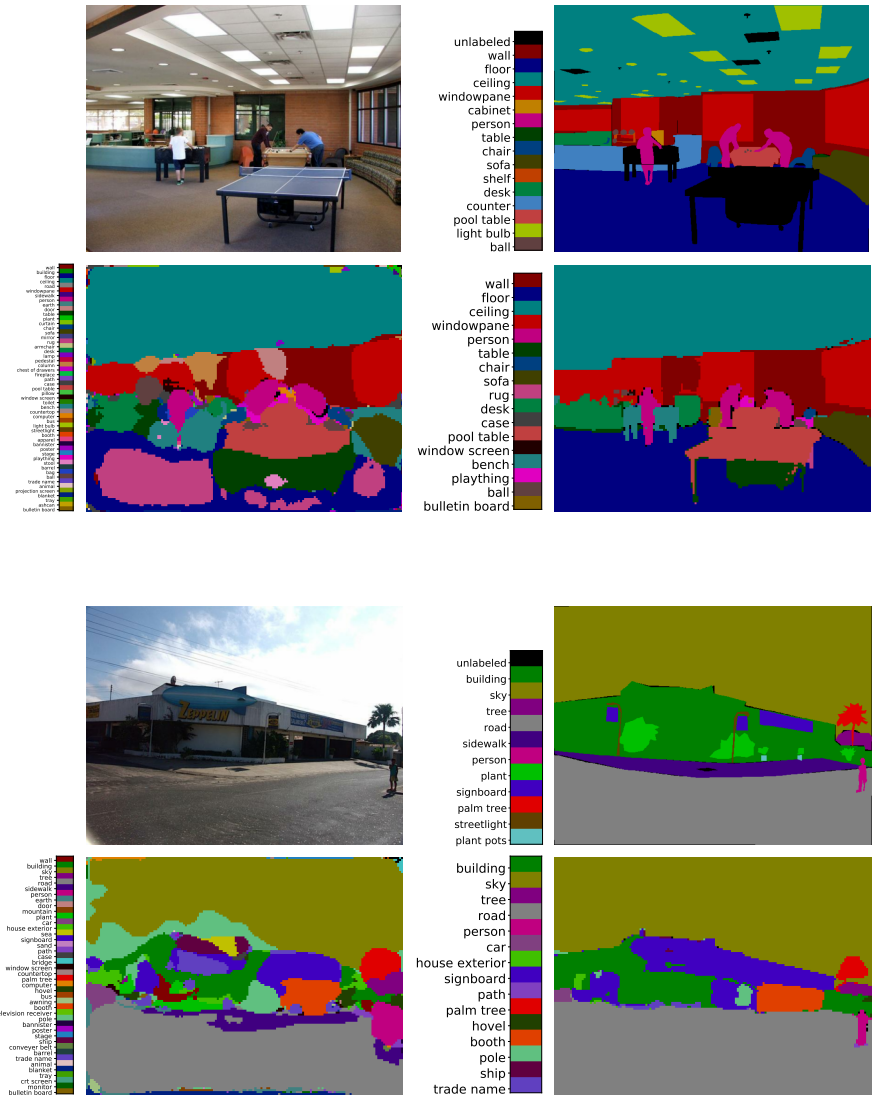

Figure 9. Predictions of OpenSeg on random examples in the `A-150` dataset (Part2).

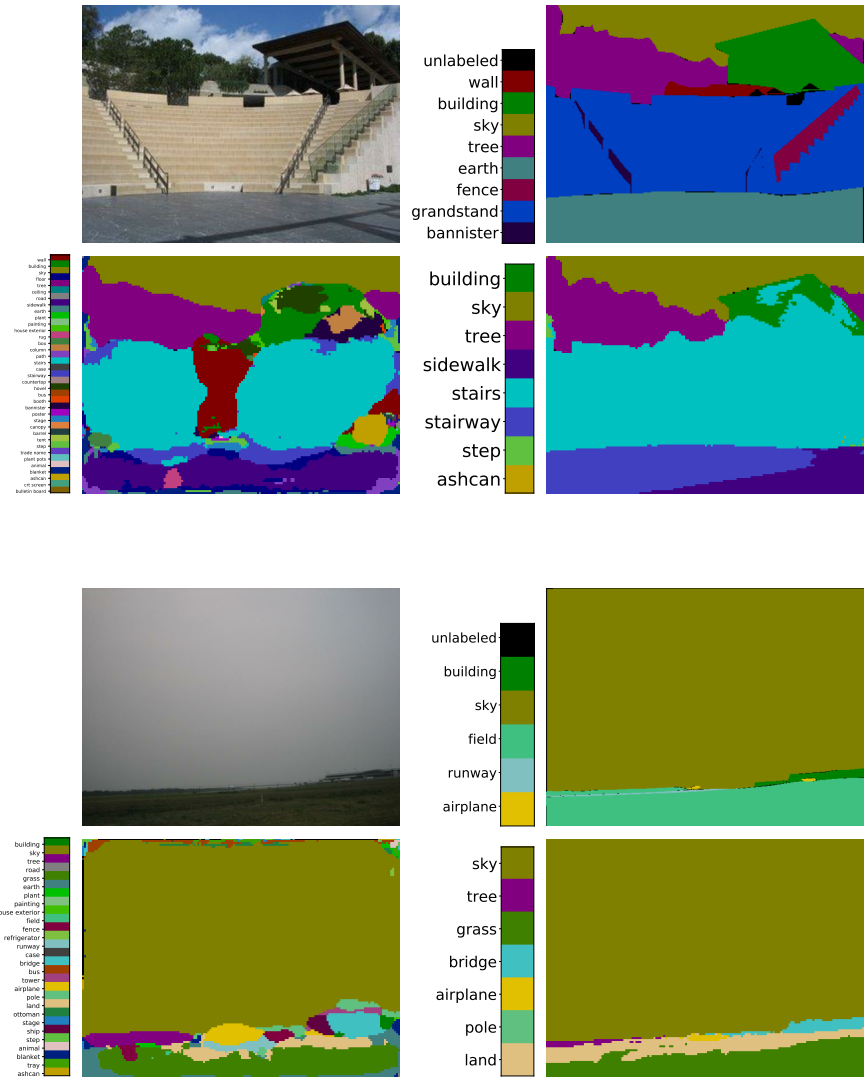

Figure 10. Predictions of OpenSeg on random examples in the `A-150` dataset (Part3).

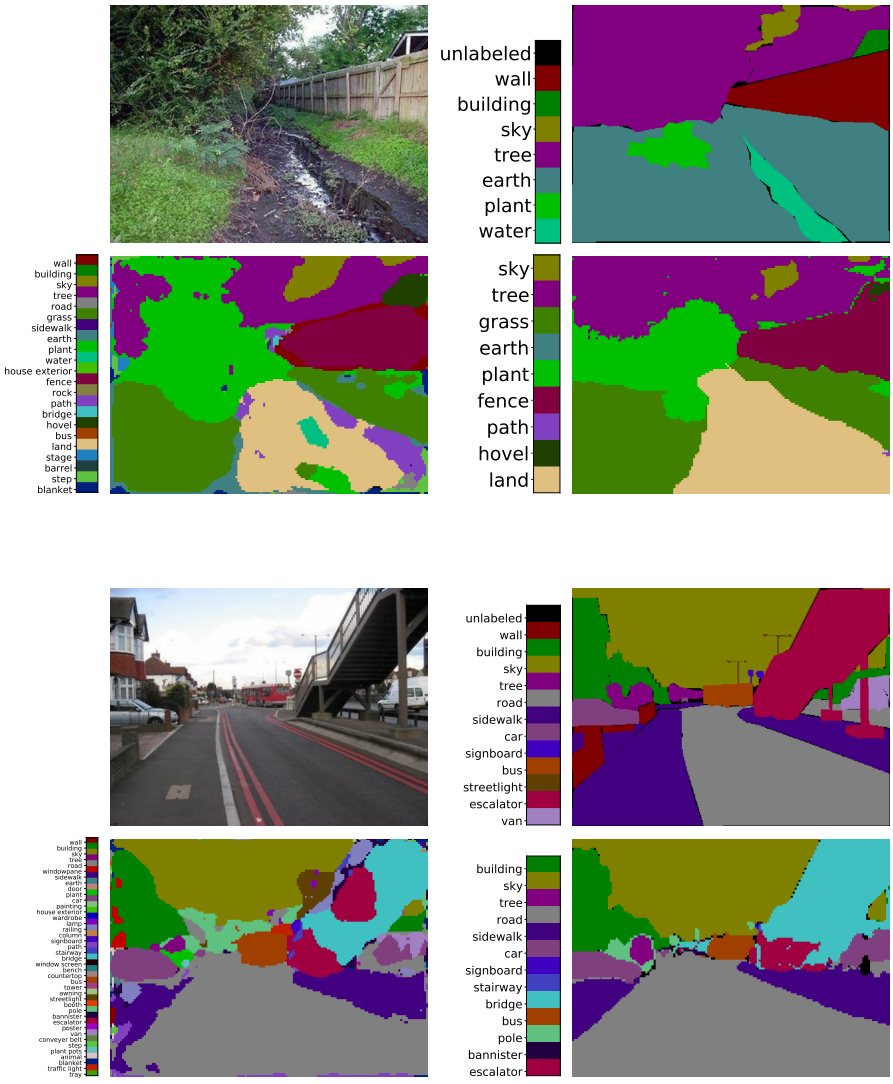

Figure 11. Predictions of OpenSeg on random examples in the `A-150` dataset (Part4).