A. Supplementary material

In this Supplementary material we consider broader impact in Sec. A.1, give more details on different foreground-background segmenters used in this work and their adaptations in Sec. A.2. We then show more qualitative results in Sec. A.3, including comparisons with other methods on the datasets used in the evaluation as well as the examples in the wild. We conclude with an in-depth analysis of the failure case of our method.

A.1. Broader impact

Semantic segmentation plays a crucial role across a wide range of fields, including healthcare, medicine, self-driving cars, and many more. While this technology can foster many applications with a positive impact, there still exists a risk of negative misuse. Additionally, since CLIP-DIY builds on foundation models that were trained on large-scale data, our method is not free from biases present in the datasets. Overall CLIP-DIY has a broad range of applications. Being training-free, CLIP-DIY can especially serve as an off-the-shelf image annotator in computing or budget-limited environments.

A.2. Foreground-background segmenters

In this section, we provide more details on the foreground-background segmenters we use in our work. We consider the two variants of FOUND [45]: we first use the coarse saliency maps produced without self-training , noted FOUND-bkg in the paper, which corresponds to the set of similar pixels to the least salient *background seed* pixel in the self-supervised feature space. We also use the quick conv1x1 model, named FOUND in the main paper, which is self-trained on only 10,553 images [49] and produces more pixel-aligned results. We refer the reader to the original paper for more details.

We also experiment with the state-of-the-art unsupervised panoptic segmentation method CutLER [50]. It produces a single mask per discovered object in a scene. We test CutLER in two different set-ups. First, since our method was designed to take one saliency map for the whole image we adapt the output of CutLER to obtain a saliency map as follows. We run CutLER per image obtaining the binary instance masks $\{\zeta_n \in [0,1]^{H\times W}\}_{n=1..N}$, with N the total number of output binary masks. We also extract the confidence scores corresponding to the masks $\{\sigma_n \in \mathbb{R}\}_{n=1..N}$. Note that the output masks are of the size of the input image. We then filter the masks and discard those with a confidence score $\sigma_n < 0.3$ similar to the value on the official CutLER repository [3]. We then aggregate the remaining masks into a saliency map M_{CUT} with:

$$M_{CUT} = \frac{1}{\mathcal{Z}} \sum_{n\in N} \sigma_n \zeta_n, \quad \text{where}$$
$$\mathcal{Z} = \max_{\text{px}} \left(\sum_{n\in N} \sigma_n \zeta_n \right) \in \mathbb{R}, \tag{5}$$

such that $M_{CUT} \in \mathbb{R}^{H\times W}$ is a normalized 2D-mask with its maximum value being 1.

Second, for a fair comparison, we also use CutLER off-the-shelf as a mask extractor. We use previously described masks $\zeta_n \in [0,1]^{H\times W}$ and with each one of them, we create an image I_n mask where the background is masked out. Each masked image I_n is then fed seperately to CLIP to obtain a CLIP prediction. We denote this approach as *CutLER mask* in Tab. 4 of the main paper.

Overall, CLIP-DIY achieves the best performances with the light self-trained FOUND as discussed in the main paper.

A.3. More qualitative results

We provide in this section more qualitative examples produced with CLIP-DIY. We first compare our method against other state-of-the-art approaches in Fig. 7 for PASCAL VOC and Fig. 8 for COCO.

In Fig. 9, we then present more in-the-wild examples and conclude this section by discussing failure cases and limitations of our method in detail.

A.3.1 Comparisons

We first present comparisons with other methods on the two segmentation datasets used in this work, namely PASCAL VOC [13] and COCO Object [26].

PASCAL VOC Fig. 7 shows randomly sampled images from PASCAL VOC dataset and the results of CLIP-DIY and our baselines. Our method produces accurate masks for all of the images and the result of CLIP-DIY is the closest to the ground truth compared to other methods. We observe that the two other methods, TCL [8] and CLIPpy [37], produce masks that are too coarse, with the latter frequently even assigning most of the image to one segment.

COCO Object Fig. 8 shows the examples from COCO dataset. While generating masks with mostly the correct category, TCL produces very noisy boundaries compared to CLIP-DIY. CLIPpy not only generates noisy masks but also produces a lot of clutter, assigning often wrong labels to background pixels.

[3] https://github.com/facebookresearch/CutLER/

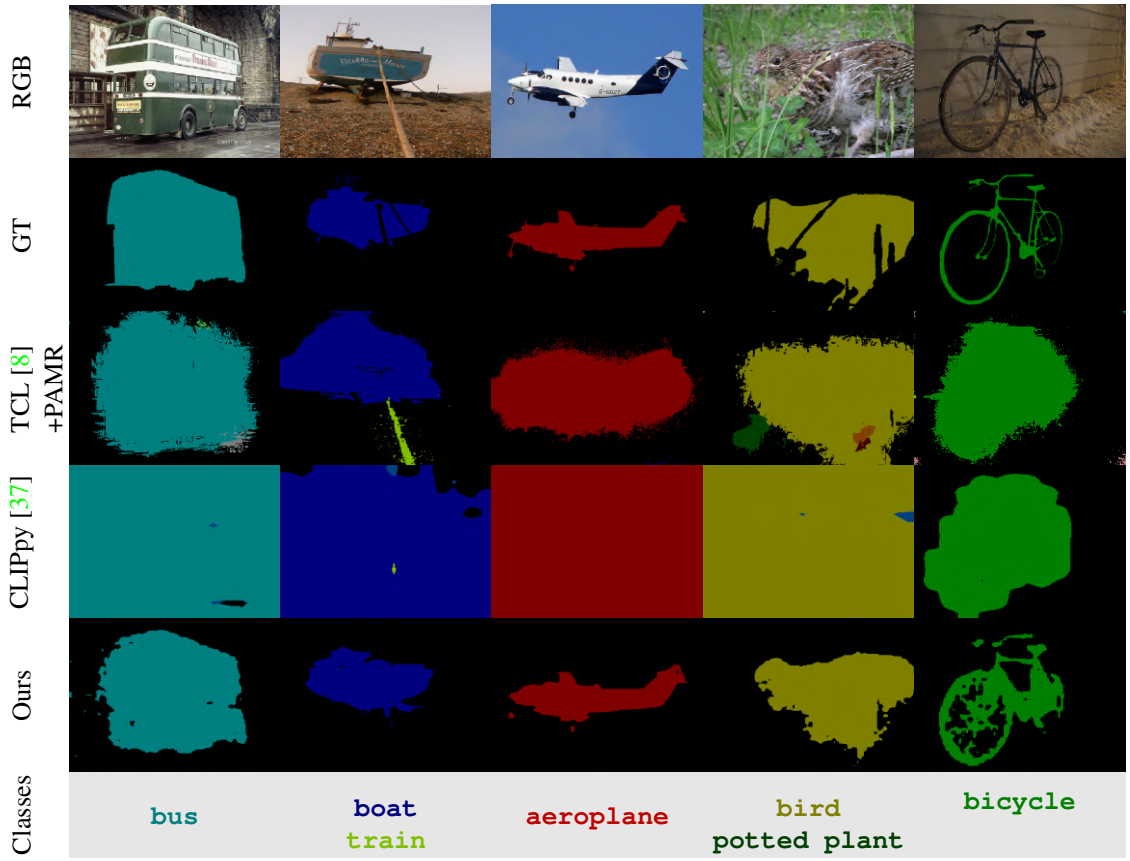

Figure 7. **Qualitative segmentation results on PASCAL VOC**. We compare our method against CLIPpy [37] and TCL (with PAMR post-processing) [8]. Our method consistently outperforms two other methods by producing accurate segmentation masks.

A.3.2 In-the-wild examples

We provide more in-the-wild examples to showcase the open-vocabulary abilities of our method. In Fig. 9 we present a couple of randomly mined images from the Web in comparison with TCL [8]. Both of the methods correctly assign queries to proper segments, even very specific types of objects, such as traditional dishes e.g. `polish dumplings` and `pasteis de nata`; monuments `Eiffel tower` and `Sacré coeur`. Moreover, thanks to CLIP backbone both methods can distinguish between different colours, e.g. `grey elephant` against `pink elephant`. However, we observe that the quality of masks produced by TCL again is not as detailed as ours. Note that TCL uses PAMR post-processing technique thus we would expect the generated masks to be more precise.

A.3.3 Failure cases

We analyze failure cases of our method in Fig. 10. We can see that `CLIP-DIY` suffers from producing incomplete masks column (a) and missing objects (c). This happens due to the saliency produced by the foreground-background segmenter, which in the case of complex, multi-object scenes focuses on certain aspects of a scene. Moreover, our method has limited performance in the case of overlapping objects, such as `dog` and `chair` in column (d). Finally, we find more failures due to inaccurate annotations, such as the one in column (b), where `bowl` is misclassified by what is inside, i.e. `carrot`.

A.3.4 Detailed quantitative results

We present detailed quantitative results on PASCAL VOC in Tab. 5 for each class. We observe that the worst performance are obtained on classes which are typically only partially visible in images, such as furniture (chair, sofa and table). This is mostly due to the decreased performance of the saliency detector in those classes, which is biased towards object-centric images. The high performance (87.6 IoU) for the `background` class confirms the efficacy of our saliency detector.

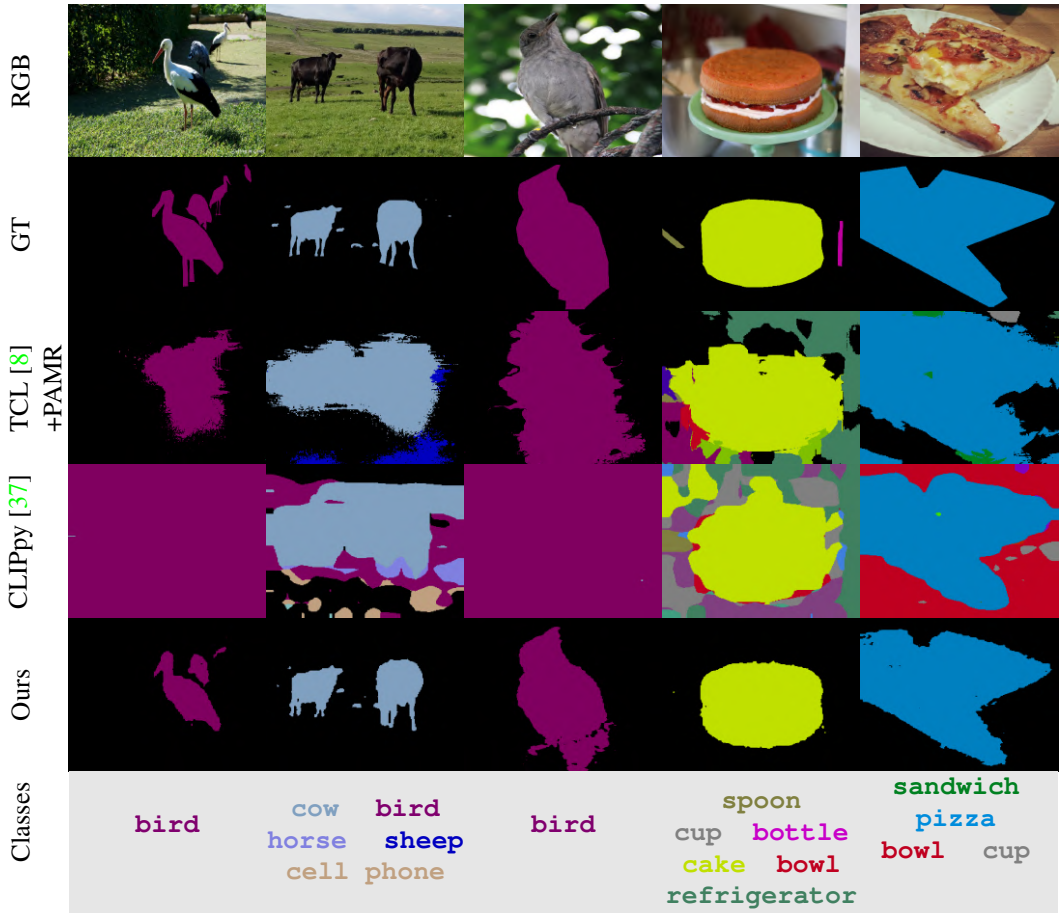

Figure 8. **Qualitative segmentation results on COCO**. We compare our method against CLIPpy [37] and TCL [8] (with PAMR [2] post-processing) [8]. Our method consistently outperforms two other methods by producing accurate segmentation masks. TCL and CLIPpy also both suffer from hallucinating or producing noisy masks.

Table 5. `CLIP-DIY` zero-shot performance (IoU) on the 21 classes from Pascal VOC. The background class is denoted as ▢.

▢	✈	🚲	🐦	🚌	🍾	🚌	🚗	🐈	🪑	🐄	🛋	🐕	🐎	🏍	🚶	🪴	🐑	🛋	🚌	🖥
87.6	78.1	33.8	77.5	62.6	65.4	71.4	66.0	81.6	16.1	75.2	20.0	78.5	69.8	63.8	56.6	37.3	79.9	25.0	68.3	37.5

Figure 9. **More examples of in-the-wild open-world segmentation.** We compare segmentation produced by our method with the results of TCL [8]. While both methods are able to detect and locate each class, including distinguishing between `pink elephant` and `grey elephant`, TCL largely over-segments objects.

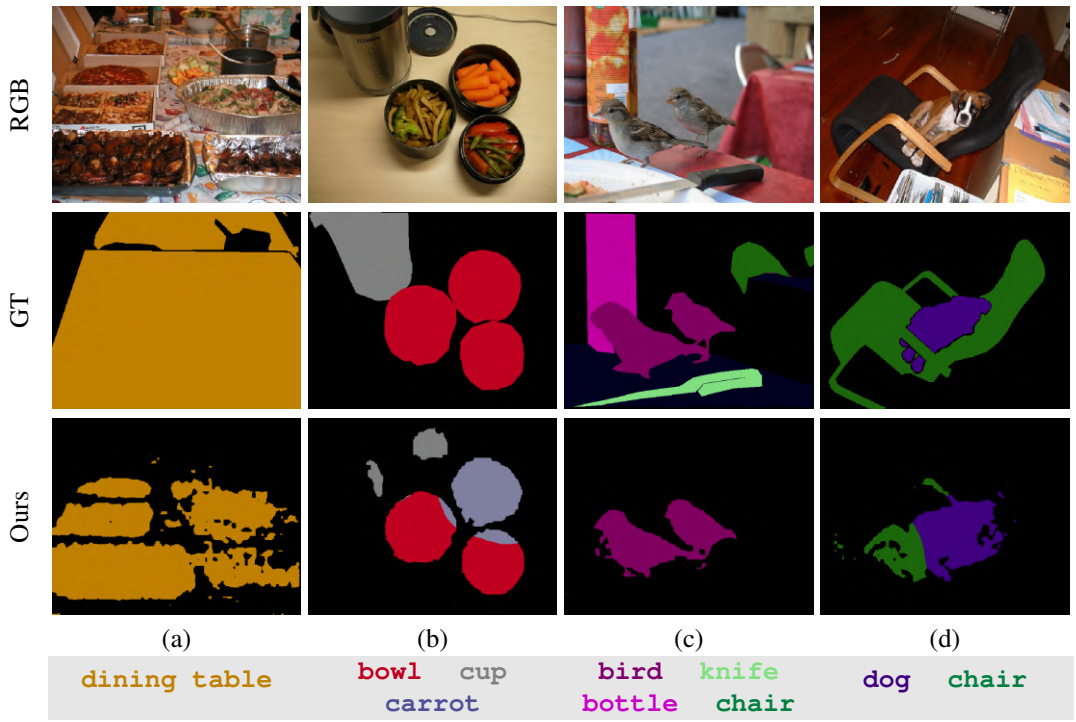

Figure 10. **Failure cases.** We show examples from both datasets. Our method at times produces incomplete masks when saliency focuses only on parts of the scene such as in (a) and (c), ambiguous classification in (b), as well confusion when classes overlap (d).