Ensemble Foreground Management for Unsupervised Object Discovery

Supplementary Material

6. Mathematical and Statistical Explanation for UnionCut

In this section, we mathematically discuss the effectiveness of UnionCut and explain why it works. Suppose $P = \{p_1, p_2, ..., p_{784}\}$ is the set of patches of an image, and F and B are subsets of P representing the set of foreground patches and background patches given by the ground truth, respectively *i.e.* $F \bigcup B = P$ and $F \bigcap B = \phi$. As for a Unit Voter (UV), its seed patch is denoted as $s_i \in P$, and the set of patches returned by the UV indicating the region similar to the seed patch s_i is defined as $\hat{F}$. With the definitions above, after the execution of a UV with s_i as the seed patch, for any background patch $\forall p_j \in B$, the probability of it being returned by the UV, *i.e.* $p_j \in \hat{F}$, can be calculated by Eq. (8).

$$
\begin{aligned}
&P(p_j \in \hat{F}|p_j \in B)\\
=&P(p_j \in \hat{F}, s_i \in F|p_j \in B) + P(p_j \in \hat{F}, s_i \in B|p_j \in B)\\
=&P(s_i \in F) \cdot P(p_j \in \hat{F}|s_i \in F, p_j \in B) + P(s_i \in B) \cdot P(p_j \in \hat{F}|s_i \in B, p_j \in B)
\end{aligned}
\tag{8}
$$

Similarly, for a foreground patch $\forall p_j \in F$, its probability of being returned by the UV is given by Eq. (9).

$$
\begin{aligned}
&P(p_j \in \hat{F}|p_j \in F)\\
=&P(p_j \in \hat{F}, s_i \in F|p_j \in F) + P(p_j \in \hat{F}, s_i \in B|p_j \in F)\\
=&P(s_i \in F) \cdot P(p_j \in \hat{F}|s_i \in F, p_j \in F) + P(s_i \in B) \cdot P(p_j \in \hat{F}|s_i \in B, p_j \in F)
\end{aligned}
\tag{9}
$$

UnionCut is required to aggregate outputs of 784 UVs to generate a heat map indicating the background area in the image, with each patch on the image being selected as the seed by one of the 784 UVs. Since there is no ground truth and prior semantic knowledge of each patch available for UnionCut and each UV is conducted on its own graph with different seed patches and anti-seed patches, each execution of a UV is independent. Therefore, the mathematical expectation of the value corresponding to a foreground patch $P_j \in F$ or background patch $P_j \in B$ on the aggregated heat map A given by Eq. (4) in the paper can be calculated by Eq. (10) or Eq. (11).

$$
E(p_j \in \hat{F}|p_j \in F) = 784 \cdot P(p_j \in \hat{F}|p_j \in F) \tag{10}
$$

$$
E(p_j \in \hat{F}|p_j \in B) = 784 \cdot P(p_j \in \hat{F}|p_j \in B) \tag{11}
$$

As introduced in Sec. 3.2, UnionCut is expected to work on an image if background regions show higher responses than the foreground on the heat map A generated by aggregating all UVs outputs. To achieve this goal, with Eq. (8), Eq. (9), Eq. (10), and Eq. (11), Inequality 12 should hold, where $P(s_i \in B)$ is of interest since it represents the prior probability that the selected seed $s_i \in P$ belongs to the background, which can be approximated by the proportion of background patches in the entire image.

Recall the statement we made in Sec. 3.2 that there are two types of UVs in UnionCut: 1) background UV: when the UV's seed patch is a background patch, the UV is expected to return a binary mask indicating the background in the image; and 2) foreground UV: when a foreground patch is used as the seed patch of a UV, the UV outputs a binary mask indicating foreground regions in the image. To solve Inequality 12 with $P(s_i \in B)$ as the variable, four items in it need to be known in advance, *i.e.* $P(p_j \in \hat{F}|s_i \in F, p_j \in B)$, $P(p_j \in \hat{F}|s_i \in B, p_j \in B)$, $P(p_j \in \hat{F}|s_i \in F, p_j \in F)$, and $P(p_j \in \hat{F}|s_i \in B, p_j \in F)$, with the meaning of each item as follows:

1) $\mathbf{P(p_j \in \hat{F}|s_i \in F, p_j \in B)}$: the probability of a background patch $p_j \in B$ being returned by a foreground UV with a foreground patch $s_i \in F$ as the seed patch;

2) $\mathbf{P(p_j \in \hat{F}|s_i \in B, p_j \in B)}$: the probability of a background patch $p_j \in B$ being returned by a background UV with a background patch $s_i \in B$ as the seed patch;

3) $\mathbf{P(p_j \in \hat{F}|s_i \in F, p_j \in F)}$: the probability of a foreground patch $p_j \in F$ being returned by a foreground UV with foreground patch $s_i \in F$ as the seed patch;

4) $\mathbf{P(p_j \in \hat{F}|s_i \in B, p_j \in F)}$: the probability of a foreground patch $p_j \in F$ being returned by a background UV with a background patch $s_i \in B$ as the seed patch.

These four items cannot be calculated directly. Therefore, we estimate their values statistically by MCE (Monte Carlo Estimation) with images in different datasets [37, 51, 64]. For example, to estimate $P(p_j \in \hat{F}|s_i \in F, p_j \in B)$, the percentage of an image's background patches being returned by each foreground UV is saved, and the mean value of these saved percentages across all images is calculated as the estimated $\hat{P}(p_j \in \hat{F}|s_i \in F, p_j \in B)$. Since $P = \{p_1, p_2, ..., p_{784}\}$ is defined as the set of patches of an image, a dataset D with K images can be denoted by $D = \{P_1, P_2, ..., P_K\}$. For the k^{th} image in the dataset, its foreground patch set and background patch set given by ground truth are represented by F_k and B_k, respectively. Besides, $\hat{F}_{kl}$ denotes the set of patches returned by a UV on the k^{th} image with $s_l \in P_k$ as the seed patch. Eq. (13) provides the MCE solution to estimate $P(p_j \in \hat{F}|s_i \in F, p_j \in B)$, $P(p_j \in \hat{F}|s_i \in B, p_j \in B)$, $P(p_j \in \hat{F}|s_i \in F, p_j \in F)$, and $P(p_j \in \hat{F}|s_i \in B, p_j \in F)$, where $|\cdot|$ calculates the number of elements in a set.

$$\begin{aligned}
&E(p_j \in \hat{F}|p_j \in B) > E(p_j \in \hat{F}|p_j \in F)\\
\Leftrightarrow &784 \cdot P(p_j \in \hat{F}|p_j \in B) > 784 \cdot P(p_j \in \hat{F}|p_j \in F)\\
\Leftrightarrow &P(p_j \in \hat{F}|p_j \in B) > P(p_j \in \hat{F}|p_j \in F)\\
\Leftrightarrow &P(s_i \in F) \cdot P(p_j \in \hat{F}|s_i \in F, p_j \in B) + P(s_i \in B) \cdot P(p_j \in \hat{F}|s_i \in B, p_j \in B) > P(s_i \in F) \cdot P(p_j \in \hat{F}|s_i \in F, p_j \in F) + P(s_i \in B) \cdot P(p_j \in \hat{F}|s_i \in B, p_j \in F)\\
\Leftrightarrow &P(s_i \in F) \cdot [P(p_j \in \hat{F}|s_i \in F, p_j \in B) - P(p_j \in \hat{F}|s_i \in F, p_j \in F)] > P(s_i \in B) \cdot [P(p_j \in \hat{F}|s_i \in B, p_j \in F) - P(p_j \in \hat{F}|s_i \in B, p_j \in B)]\\
\Leftrightarrow &[1 - P(s_i \in B)] \cdot [P(p_j \in \hat{F}|s_i \in F, p_j \in B) - P(p_j \in \hat{F}|s_i \in F, p_j \in F)] > P(s_i \in B) \cdot [P(p_j \in \hat{F}|s_i \in B, p_j \in F) - P(p_j \in \hat{F}|s_i \in B, p_j \in B)]\\
\Leftrightarrow &P(p_j \in \hat{F}|s_i \in F, p_j \in B) - P(p_j \in \hat{F}|s_i \in F, p_j \in F) > P(s_i \in B) \cdot [P(p_j \in \hat{F}|s_i \in B, p_j \in F) - P(p_j \in \hat{F}|s_i \in B, p_j \in B) + P(p_j \in \hat{F}|s_i \in F, p_j \in B) - P(p_j \in \hat{F}|s_i \in F, p_j \in F)]
\end{aligned} \tag{12}$$

Dataset	$\hat{P}(p_j \in \hat{F}\|s_i \in F, p_j \in B)$	$\hat{P}(p_j \in \hat{F}\|s_i \in B, p_j \in B)$	$\hat{P}(p_j \in \hat{F}\|s_i \in F, p_j \in F)$	$\hat{P}(p_j \in \hat{F}\|s_i \in B, p_j \in F)$
COCO2014 train [37]	0.1215	0.5496	0.1981	0.0563
COCO20K [37, 53]	0.1203	0.5499	0.1987	0.0572
ECSSD [51]	0.0308	0.7270	0.1927	0.0198
DUTS-TR [64]	0.0495	0.7628	0.2098	0.0264
DUTS-TE [64]	0.0549	0.6621	0.2099	0.0294

Table 7. **MCE results of different datasets for UV.**

$$\begin{aligned}
\hat{P}(p_j \in \hat{F}|s_i \in F, p_j \in B) &= \frac{\sum_{k=1}^{K} \sum_{s_l \in F_k} \frac{|\hat{F}_{kl} \cap B_k|}{|B_k|}}{\sum_{k=1}^{K} \mid F_k \mid}\\
\hat{P}(p_j \in \hat{F}|s_i \in B, p_j \in B) &= \frac{\sum_{k=1}^{K} \sum_{s_l \in B_k} \frac{|\hat{F}_{kl} \cap B_k|}{|B_k|}}{\sum_{k=1}^{K} \mid B_k \mid}\\
\hat{P}(p_j \in \hat{F}|s_i \in F, p_j \in F) &= \frac{\sum_{k=1}^{K} \sum_{s_l \in F_k} \frac{|\hat{F}_{kl} \cap F_k|}{|F_k|}}{\sum_{k=1}^{K} \mid F_k \mid}\\
\hat{P}(p_j \in \hat{F}|s_i \in B, p_j \in F) &= \frac{\sum_{k=1}^{K} \sum_{s_l \in B_k} \frac{|\hat{F}_{kl} \cap F_k|}{|F_k|}}{\sum_{k=1}^{K} \mid B_k \mid}
\end{aligned} \tag{13}$$

Tab. 7 illustrates the estimation of the four items in Inequality 12 with Eq. (13) on different datasets. It can be observed that the probability (*i.e.* $\hat{P}(p_j \in \hat{F}|s_i \in F, p_j \in B)$) of a background patch being returned by a foreground UV is very low, and vice versa (*i.e.* $\hat{P}(p_j \in \hat{F}|s_i \in B, p_j \in F)$). However, there is a significant difference in the probability (*i.e.* $\hat{P}(p_j \in \hat{F}|s_i \in F, p_j \in F)$) of foreground patches being returned by a foreground UV, compared to the probability (*i.e.* $\hat{P}(p_j \in \hat{F}|s_i \in B, p_j \in B)$) of background patches being returned by a background UV. Specifically, when a background patch is selected, other background patches are more likely to be segmented, whereas when a foreground patch is selected, other foreground patches are not as easily segmented. In other words, during the voting process, background UVs are relatively active—they, while refusing to vote for foreground patches, nominate most of the background patches. In contrast, foreground UVs are relatively inactive: they refuse to vote for background patches, and also only nominate a small portion of the foreground patches. This leads to UnionCut's aggregated final voting results showing that background patches receive significantly more votes than foreground patches, even in extreme cases—for instance, when the number of background UVs is smaller than that of foreground UVs (i.e., when the foreground occupies the majority of the image), this situation still holds. This difference between the two types of UVs is counterintuitive. but precisely, this critical distinction allows UnionCut to work robustly, even in images where the background occupies a tiny portion.

Based on the above findings, we can explain why UnionCut works robustly for the detection of foreground unions with ensemble learning theories.

UnionCut's Effectiveness and Robustness Assuming that we regard all UVs as weak classifiers for background classification (regardless of whether they are foreground UVs or background UVs), meaning that the regions with a value of 1 in all UVs'output binary masks are considered background, then a background UV can always correctly classify the background. For a foreground UV, it can be interpreted here as a background weak classifier handling hard cases: although it may erroneously return foreground as background, its less active nature and its tendency to suppress both foreground and background in its output mean that the errors it produces have minimal impact on the final voting result. Therefore, in the final aggregated result of UnionCut, the correct background segmentation produced by the background UVs dominates, causing UnionCut as a strong classifier to reliably output the background regions of the image, which, after being inverted and thresholded, become the final foreground union output by UnionCut.

Impact of the Proportion of the Background in the image on UnionCut With MCE results in Tab. 7, Inequality 12 can be solved by substituting the estimated values in Tab. 7 into Inequality 12. As shown in Tab. 9, taking COCO2014 train [37] as an example, Inequality 12 holds when $P(s_i \in B) > 0.1344$. Since $P(s_i \in B)$ in an im-

Dataset	$\hat{P}(p_j \in \hat{F} \mid s_i \in F, p_j \in B)$	$\hat{P}(p_j \in \hat{F} \mid s_i \in B, p_j \in B)$	$\hat{P}(p_j \in \hat{F} \mid s_i \in F, p_j \in F)$	$\hat{P}(p_j \in \hat{F} \mid s_i \in B, p_j \in F)$
COCO2014 train [37]	0.536	0.8832	0.6554	0.3962
COCO20K [37, 53]	0.536	0.8832	0.6554	0.3977
ECSSD [51]	0.3348	0.9467	0.7131	0.2668
DUTS-TR [64]	0.3567	0.9568	0.6966	0.2853
DUTS-TE [64]	0.3518	0.9183	0.7195	0.2476

Table 8. **MCE results of different datasets for cosine similarity matching.**

age is approximated by the percentage of area occupied by background patches, $P(s_i \in B) > 0.1344$ can be understood as indicating that when the background occupies more than 13.44% of the image area, UnionCut can robustly detect the foreground union in the image. In other words, even if there are large objects in an image, UnionCut works robustly if the foreground union occupies less than 86.56% of the image.

Dataset	Solution
COCO2014 train [37]	$P(s_i \in B) > 0.1344$
COCO20K [37, 53]	$P(s_i \in B) > 0.1372$
ECSSD [51]	$P(s_i \in B) > 0.1862$
DUTS-TR [64]	$P(s_i \in B) > 0.1787$
DUTS-TE [64]	$P(s_i \in B) > 0.1967$

Table 9. **Solutions of Inequality 12 for UV based on the MCE results in Tab. 7**

7. Examples of Foreground Union Detection on Images with Large Foreground

In this section, we show UnionCut and UnionSeg's successful examples on images whose background is smaller than the foreground to support our claim in Sec. 3.2 that UnionCut can stay effective on images of large foreground areas. As shown in Fig. 6, the foreground union of three images occupied mainly by the foreground are accurately detected by both UnionCut and UnionSeg, demonstrating that UnionCut and UnionSeg stay effective on images of small backgrounds.

Dataset	Solution
COCO2014 train [37]	$P(s_i \in B) > 0.1968$
COCO20K [37, 53]	$P(s_i \in B) > 0.1973$
ECSSD [51]	$P(s_i \in B) > 0.3574$
DUTS-TR [64]	$P(s_i \in B) > 0.3361$
DUTS-TE [64]	$P(s_i \in B) > 0.3541$

Table 10. **Solutions of Inequality 12 for consine similarity matching based on the MCE results in Tab. 8**

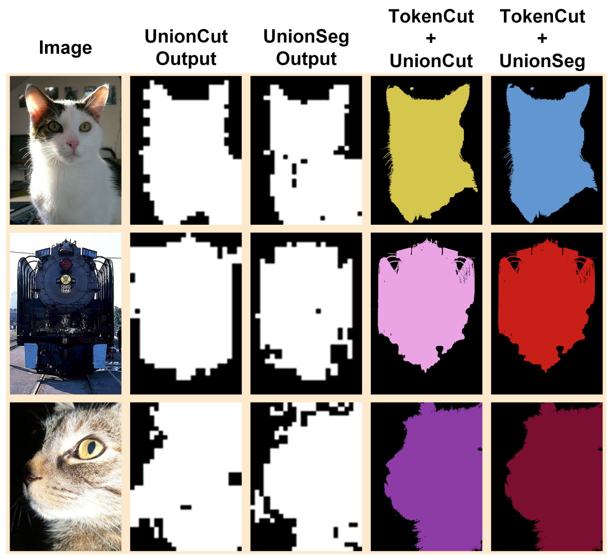


Figure 6. **Examples of UnionCut and UnionSeg's foreground union detection results on images with large foreground**.

8. Comparison between UV and Straitforward Feature Matching

In this section, we continue to use the mathematical and statistical methods in Appendix 6 to demonstrate the unique advantages and necessity of UV, compared with other straightforward feature matching methods. Here, we make matching similar patches with cosine similarity used by [53, 54, 59] as an example. Specifically, given the seed patch $s_i \in P$ and its feature vector $k_i \in K$, the patches sharing features similar to s_i can be obtained and defined as $\hat{F} = \{p_j \mid p_j \in P, k_j^T k_i > 0\}$. Based on Eq. (13), $P(p_j \in \hat{F} \mid s_i \in F, p_j \in B)$, $P(p_j \in \hat{F} \mid s_i \in B, p_j \in B)$, $P(p_j \in \hat{F} \mid s_i \in F, p_j \in F)$, and $P(p_j \in \hat{F} \mid s_i \in B, p_j \in F)$ of cosine similarity matching can be estimated, as shown in Tab. 8. Compared with UV, when a foreground patch is selected as the seed patch, the probability (*i.e.* $P(p_j \in \hat{F} \mid s_i \in F, p_j \in B)$) of a background patch being returned is much larger, and vice versa, indicating that consine similarity feature matching is more likely to make mistakes as a weak classifier than UV. Besides, the difference between $P(p_j \in \hat{F} \mid s_i \in B, p_j \in B)$ and $P(p_j \in \hat{F} \mid s_i \in F, p_j \in F)$

is not significant, indicating that cosine similarity matching of a foreground seed patch and background seed patch are similarly active. This reduces the dominant influence of the cosine similarity of the background seed patches on the final aggregated voting results.

Furthermore, we can also solve Inequality 12 with the estimated results in Tab. 8. As shown in Tab. 10, taking DUTS-TE as an example, the aggregated strong classifier with cosine similarity matching as weak classifiers only works on images with the background area occupies over 35.41% (compared with 19.67% for UV), *i.e.* the foreground should occupy less than 64.59% areas of the image (compared with 80.33% for UV), which much limits the robustness and effectiveness of the strong classifier since there are many image cases that break this constraint, examples can be seen in Fig. 6.

Weak Classifier's Diversity We can also compare UV and cosine similarity matching in terms of the requirement of ensemble methods with ensemble learning theories. Specifically, to obtain a robust enough strong classifier, there is a diversity requirement for weak classifiers, *i.e.* weak classifiers should be different from each other and as diverse as possible [18]. A large number of homogeneous weak classifiers can cause the performance of the aggregated strong classifier to degrade to that of a weak classifier.

Compared to UV, using straightforward cosine similarity matching of a seed patch as a weak classifier leads to a large number of homogeneous weak classifiers. That is because directly computing the cosine similarity between a seed patch's feature and those of other patches requires calculating the distance matrix of all patches, and this distance matrix is symmetric. This symmetry means that if a patch $p_j \in P$ appears in the cosine similarity matching results when using another patch $p_i \in P$ as the seed patch, then p_i will also appear in the cosine similarity matching results when using p_j as the seed patch. Ultimately, this results in a certain homogeneity in the output of the weak classifiers.

In contrast, UV is more diverse as a weak classifier because the output of each UV of a seed patch $s_i \in P$ depends not only on the features of s_i but also on the features of its anti-seed patches $B_f = \{p_b | p_b \in P, b \neq f, k_b^T k_f < 0\}$. Since the anti-seed patches for each seed patch s_i are different, the introduction of anti-seed patches eliminates the previously mentioned symmetry, making UV as a weak classifier more varied, leading to UnionCut as a strong classifier being more robust.

In summary, based on the previous discussion with Tab. 7, Tab. 8, Tab. 9 and Tab. 10, our proposed UV has advantages in robustness over straightforward feature matching, and is less limited by the impact of large area occupied by the foreground. Moreover, although our UV is more complex than straightforward feature matching, its pipeline, which models the image as a graph, can effectively consider both the seed patch and the anti-seed patches during execution, which can hardly be done by straightforward feature matching. This makes our UV technically irreplaceable (this does not mean that no other algorithm can achieve the same effect as the UV, but more research is needed to explore this possibility).

9. Reliability of Corner Prior

To explore the reliability of the corner prior used by UnionCut, we focus on its success rate on images from different datasets [37, 51, 64, 74]. Specifically, we calculate the success rate by assessing the proportion of images in each dataset where the union of the ground truth occupies less than four corners of the image. As shown in Tab. 11, the corner prior's success rate achieves over 99% on every selected dataset, indicating its robustness in checking the reliability of the foreground union mask detected by UnionCut.

Dataset	Success Rate
COCO2014 train [37]	99.92%
COCO2017 train [37]	99.92%
COCO20K [37, 53]	99.92%
ECSSD [51]	100%
DUTS-TR [64]	99.83%
DUTS-TE [64]	99.89%
DUTS-OMRON [74]	100%

Table 11. **Successful rate of the corner prior utilized by UnionCut with different datasets.**

10. FOUND vs. UnionSeg

Fig. 7 compares the pipeline of FOUND [54] and UnionSeg. In [54], an unnamed UOD method, here referred to as "FOUND-", is proposed to conduct UOD on the dataset DUTS-TR [64]. After that, the object discovered by FOUND- is used as pseudo-labels to train a ViT surrogate model (*i.e.* FOUND). Similarities and differences between FOUND and UnionSeg are listed below:
Similarities: they share the same model structure.
Differences:
1) their loss functions and training settings are different;
2) Their pseudo-labels have different sources and represent different meanings: the pseudo-labels for FOUND are generated by FOUND- and represent the objects discovered in the image (which do not necessarily cover the majority of objects in the image). In contrast, UnionSeg's pseudo-labels are generated by UnionCut and are designed to cover most of the object regions in the image, i.e., the foreground union;
3) Based on the difference mentioned above, the function of UnionSeg and FOUND are also different: FOUND and

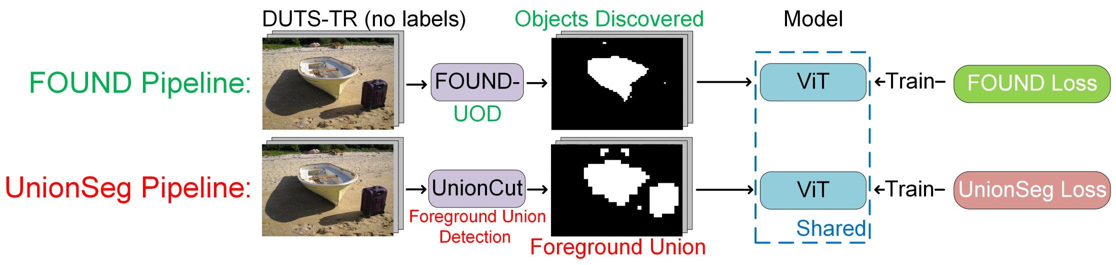
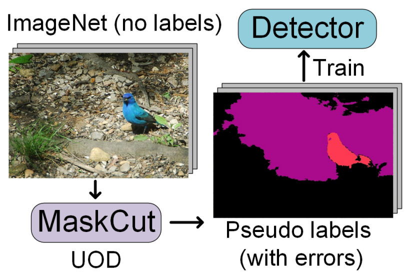
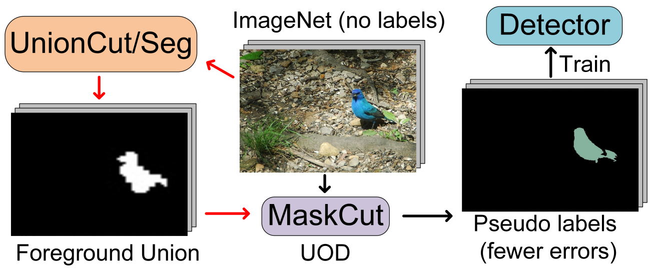


Figure 7. The comparison of the framework between FOUND [54] and UnionSeg.



(a) Pipeline of CutLER



(b) Pipeline of CutLER+UnionCut/UnionSeg

Figure 8. The comparison between the pipeline of CutLER and CutLER+UnionCut/UnionSeg.

FOUND- are proposed for unsupervised object discovery (*i.e.* UOD), while UnionSeg and UnionCut are utilized to predict the union of foreground area (*i.e.* foreground union) in an image, which are used as foreground priors for UOD methods.

11. Combining UnionCut/UnionSeg with Existing UOD methods and Implementation Details

The core idea of combining UnionCut or UnionSeg with existing UOD methods is to replace their default foreground priors with UnionCut/UnionSeg. In this section, we introduce how to apply UnionCut/UnionSeg to existing UOD methods. Note that the methods designed by us are not the only way to combine UnionCut/UnionSeg with existing UOD methods. We welcome the proposal of more advanced combining methods to further enhance UnionCut/UnionSeg's boosting effect on UOD algorithms. More technical details of this section can be seen in our code.

11.1. Basic Usage of UnionCut and UnionSeg

UnionCut and UnionSeg can be integrated into UOD algorithms in various ways; here, we show their basic usage. Assuming the foreground union U detected by UnionCut or UnionSeg as the ground truth, the precision of a mask discovered by UOD algorithms can be calculated by Eq. (14)

$$Precision(mask, U) = \frac{Area(mask \cap U)}{Area(mask)} \tag{14}$$

where $Area(\cdot)$ returns the area of a mask, based on which UnionCut and UnionSeg can be utilized in two ways:
1) For judging whether a discovered area belongs to the foreground, an area will be considered as part of the foreground in the image if its precision is higher than a predetermined threshold θ (*e.g.* 0.5);
2) To determine when to stop further discovery, exploration stops if the majority (a percentage γ, *e.g.* 80% area) of the foreground union given by UnionCut or UnionSeg has been discovered.

In practice, we recommend setting $\theta = 0.5$ and $\gamma = 0.8$ when using the basic usage of UnionCut or UnionSeg.

11.2. LOST+UnionCut/UnionSeg

LOST [53]'s default foreground prior is as follows: An image is divided into patches corresponding to the serialized input of a ViT. The image is organized into a graph with each patch as a node. For any two patches in the graph, they are connected by links if their cosine similarity is more than 0. Then all patches are sorted by their degrees (*i.e.* the number of links connected to a patch) in ascending order, and the first patch after being sorted is made as the foreground seed based on the assumption made by Siméoni *et al.* that the area occupied by the foreground should be smaller than the background [53]. Then a discovered object region is expanded from the foreground seed across the entire image. More details can be found in [53].

In this paper, we combine LOST and Union-

Cut/UnionSeg by limiting the foreground seed selection range within the foreground union output by UnionCut/UnionSeg, *i.e.* a patch will not be chosen as the seed if it is not in the foreground union, even though its degree is minimal. Besides, the discovered object area is expanded from the seed across the region of the foreground union instead of the entire image.

11.3. TokenCut+UnionCut/UnionSeg

TokenCut is a single-object discovery method that divides an image into an object area and background (may contain other objects). To combine TokenCut with UnionCut/UnionSeg, we replace its default foreground prior with our approach, selecting the area with higher precision (Eq. (14)) as the discovered object from the bipartition.

11.4. FOUND+UnionCut/UnionSeg

As introduced in Sec. 10, FOUND is trained based on the output of FOUND- as pseudo-lables. We combine UnionSeg/UnionCut with FOUND by making the FOUND trained based on the intersection between FOUND-'s output and foreground union by UnionSeg/UnionCut as the pseudo-labels.

11.5. CutLER+UnionCut/UnionSeg

11.5.1. The pipeline of combining CutLER and UnionCut or UnionSeg

CutLER detects multiple objects in three steps, as shown in Fig. 8a: 1) MaskCut [69] (a UOD method) discovers objects up to three times per image to generate pseudo-annotations on ImageNet [17]; 2) training a Cascade Mask-RCNN [7] with these pseudo-annotations; 3) using the trained Cascade Mask-RCNN to update pseudo-annotations and retraining the model. As depicted in Fig. 8b, we use UnionCut/UniongSeg to eliminate errors from MaskCut, enhancing the detector's performance after training. Specifically, UnionCut/UniongSeg is integrated with CutLER at its first step, *i.e.* MaskCut. Without a foreground prior, MaskCut's excessive discovery leads to misidentifying the background as the foreground, so Wang *et al.* [69] limits MaskCut to three discoveries per image. Using UnionCut/UniongSeg, we can remove this balance and make MaskCut stop when 80% of the foreground union given by UnionCut/UniongSeg is detected, discarding discovered areas with precision (Eq. (14)) below 0.5. In the paper, all results related to CutLER and CutLER+UnionCut/UniongSeg are performances of the Cascade Mask-RCNN after training.

11.5.2. Training Details of CutLER+UnionSeg

We use the official implementation of CutLER [69] in our experiments, using our implementation of MaskCut [69] with UnionSeg to replace the initial MaskCut in the original CutLER. Following CutLER, a ViT-B/8 [19] pretrained by DINO [9] is used as the image feature extractor for MaskCut. In terms of training CutLER+UnionSeg, first, we make our MaskCut+UnionSeg to conduct object discovery on all images (1.28 million) from ImageNet. Due to the large number of images to be processed, although our algorithm can judge when to stop discovery, we additionally made MaskCut+UnionSeg conduct up to 5 times exploration per image. With 18 processes running simultaneously, it took us 3 weeks to generate UOD results for all images, which were used as pseudo-annotations of instance segmentation for the dataset. Then, we used ImageNet and these pseudo-annotations to train a Cascade Mask-RCNN model with the official implementation of CutLER for 10,000 iterations with a learning rate of 0.01 and weight decay of 0.001 as a warm-up. After that, the trained model was used to update pseudo-annotations for images in ImageNet. We further trained the model for 60,000 external iterations with the updated pseudo-annotations, using a learning rate of 0.005 and weight decay of 0.0001. All other settings, *e.g.* optimizer, batch size and DropLoss threshold, were inherited entirely from the official CutLER.

12. Performance Upper Boundary of UOD Methods Designed for Multiple Objects Discovery on Single Object Discovery

Single object discovery requires UOD algorithms to predict only one bounding box for an image, and CorLoc is calculated by checking if the predicted bounding box matches any one object's annotation in the image. However, when applying UOD methods able to predict multiple objects per image to single object discovery, strategies of selecting only one predicted bounding box are required to filter out redundant predictions, *e.g.* selecting the predicted bounding box of the largest area or highest confidence. However, various strategies lead to different performances [54]. As such, Rambhatla *et al.* [45] proposed to use average best overlap [58] to evaluate the performance upper boundary of UOD algorithms designed for multiple object discovery on single object discovery. Specifically, each predicted bounding box is compared with all ground truth bounding boxes in an image. If any prediction matches a ground truth bounding box, the UOD algorithm will be considered successful on this image. The essence of this approach is the assumption that an optimal strategy exists to select the predicted box from a set of predictions which is most likely to match the ground truth. Applying this assumed strategy allows for measuring the UOD algorithm's performance on single object discovery in an ideal scenario, *i.e.* its upper boundary performance. We also try to apply this strategy to evaluate the upper-boundary performance of CutLER (with or without UnionSeg) on single object discovery. As shown in Tab. 12, after considering the upper boundary, the performance's upper boundary of CutLER is also increased by UnionSeg and achieves state-of-the-art performance on all three benchmarks, indicating the effectiveness of our proposed UnionSeg.

Method	UB	VOC07	VOC12	COCO20K
- No learning -				
Selective Search [58]		18.8	20.9	16.0
EdgeBoxes [80]		31.1	31.6	28.8
Kim *et al.* [30]		43.9	46.4	35.1
Zhang *et al.* [75]		46.2	50.5	34.8
DDT+ [70]		50.2	53.1	38.2
rOSD [61]		54.5	55.3	48.5
LOD [62]		53.6	55.1	48.5
DINO-seg [9, 53](ViT-S/16 [9])		45.8	46.2	42.0
LOST [53](ViT-S/16 [9])		61.9	64.0	50.7
DSS [40](ViT-S/16 [9])		62.7	66.4	52.2
MOST [45]	✓	74.8	77.4	67.1
TokenCut [21](ViT-S/16 [9])		68.8	72.1	58.8
TokenCut(ViT-S/16 [9])+UnionCut		69.2(**0.4↑**)	72.3(**0.2↑**)	62.1(**3.3↑**)
TokenCut(ViT-S/16 [9])+UnionSeg		69.7(**0.9↑**)	72.7(**0.6↑**)	62.6(**3.8↑**)
- With learning -				
FreeSOLO [54, 67]		44.0	49.7	35.2
LOD+CAD [21, 53]		56.3	61.6	52.7
rOSD+CAD [21, 53]		58.3	62.3	53.0
LOST+CAD [53](ViT-S/16 [9])		65.7	70.4	57.5
SelfMask [52]		72.3	75.3	62.7
FOUND [54](ViT-S/16 [9])		72.5	76.1	62.9
CutLER [69](ViT-B/8 [9])†		73.3	69.5	70.7
CutLER(ViT-B/8 [9])+UnionSeg		73.8(**0.5↑**)	71.2(**1.7↑**)	72.4(**1.7↑**)
CutLER [69](ViT-B/8 [9])†	✓	87.3	84.3	89.3
CutLER(ViT-B/8 [9])+UnionSeg	✓	87.4(**0.1↑**)	85.1(**0.8↑**)	89.3

Table 12. **Performances of UOD methods on single object discovery, with CorLoc as the metric and the upper boundary of UOD methods detecting multiple objects considered.** UB: the short for Upper Boundary. CAD: a class-agnostic detector trained with the output of no-learning UOD methods as the ground truth. †: results of official implementations and checkpoints by our measurement.

13. UnionCut's Effectiveness on Self-supervised Instance Segmentation

In this section, CutLER (MaskCut) and TokenCut are combined with UnionCut before being used to generate pseudo-labels for images in the dataset. After that, these pseudo-labels are used to train a class-agnostic SOLOv2 [65] model. The subset (2913 images) of VOC2012 [22] where images are with pixel-level annotations are used as the benchmark for training and evaluating the performance of the model [65] trained with pseudo-labels given by different UOD algorithms. Five folds are used for cross-validation. Specifically, for each fold, the dataset is divided into training, validation, and test sets in a ratio of 8:1:1. The model is trained with images of the training part and corresponding pseudo-labels generated by a UOD method. The model's weights are saved for every 1000 iterations. The validation set and handcrafted ground truth provided by VOC2012 are used to evaluate the model's weights of each iteration. The weights that perform best on the validation set are selected,

Method	$AP_{0.5}^{mask}$	$AP_{0.5}^{box}$
LOST†	10.2	14.9
MaskDistill‡	11.3	16.2
TokenCut†	19.0	22.8
TokenCut+UnionCut	**19.2(0.2↑)**	**22.9(0.1↑)**
CutLER†	18.5	22.0
CutLER+UnionCut	**20.9(2.4↑)**	**23.5(1.5↑)**

Table 13. **Performance reported on VOC12 of SOLOv2[65] trained with pseudo-labels given by different UOD methods.** †: codes from the official implementation. ‡: our replication based on the paper.

whose performance is evaluated with the test set with handmade ground truth and reported here. All training in this experiment uses a fixed random seed of 3407 and trains the model for $30,000$ iterations per fold. Models' parameters are updated using an Adam optimizer [31] with a learning rate 0.0001 and a mini-batch size 16. Average precision (AP) is used as the metric. Consistent experimental setups apply to all other UOD methods. Tab. 13 shows that UnionCut improves the quality of pseudo-labels given by MaskCut and TokenCut, indicating its effectiveness in boosting the performance of UOD methods as a robust foreground prior.

Note that unlike Tab. 3 in Sec. 4.3 where ImageNet (1.28 million images) is used as the training set, considering the computational cost of UnionCut, only VOC12 (2983 images with pixel-level annotations) is utilized for training the model in this experiment, which is much fewer than ImageNet. Since UnionSeg's effectiveness has been discussed in Sec. 4.3 in terms of instance segmentation, this section aims to show the effectiveness of UnionCut additionally and is not for making the model achieve state-of-the-art performance.

14. Examples of Images without Foreground Union Fully Annotated

This section provides example images from VOC12, where the union of the ground truth does not fully cover the foreground union. As shown in Fig. 9, the keyboard, stereo, cup, etc., are not labelled by the ground truth (the left column); the vase and light are not labelled (the mid column); and the end table is not labelled (the right column).

15. Qualitative Analysis

In this section, we visualize and analyse how UnionCut and UnionSeg boost the performance of our selected baseline UOD algorithms, *i.e.* TokenCut and MaskCut, and provide more visualization.

Foreground Judgement UnionCut/UnionSeg enables UOD methods to judge if a discovered area belongs to the

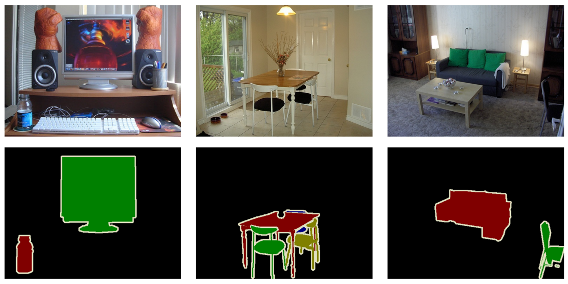


Figure 9. **Visualization of images in VOC12 whose ground truth union do not cover the foreground union of the image.**

foreground. Taking TokenCut as an example, as shown in Fig. 10, TokenCut initially chooses the background part from the bipartition as the discovered result. With UnionCut or UnionSeg, the segmentation mostly covered by the foreground union is chosen to correct TokenCut's error.

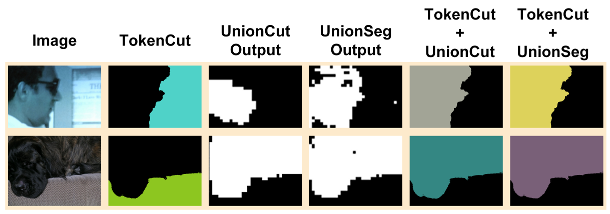


Figure 10. TokenCut's errors fixed by UnionCut and UnionSeg.

Complete Discovery UnionCut/UnionSeg enables UOD algorithms, especially those detecting multiple objects, to judge when to stop further discovery without under or over-discovery. Fig. 11 illustrates two examples where UnionCut and UnionSeg help MaskCut remove a misidentified background area (the 1st row) and prevent missing one object (the car on the left of the 2nd image) by ensuring the majority of the foreground union is discovered.

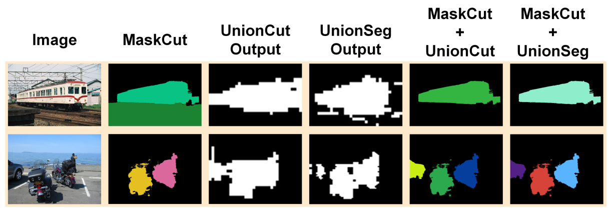


Figure 11. Examples of removing MaskCut's detection errors and making its discovery stop at appropriate time.

Additional Visualization Fig. 12 provides additional qualitative results of different UOD methods, indicating that UnionCut and UnionSeg detect foreground union accurately, fix errors made by MaskCut without missing objects or including background, and, as such, improve the performance of MaskCut. Only MaskCut here (with or without UnionSeg) conducts discovery multiple times, while others conduct UOD one time per image. We made MaskCut conduct up to 3 times discovery per image following the recommendation of the original work [69]. As for our MaskCut+UnionSeg, to show that UnionSeg enables UOD algorithms to stop discovery at the appropriate time, we made MaskCut+UnionSeg conduct up to 50 times discovery per image.

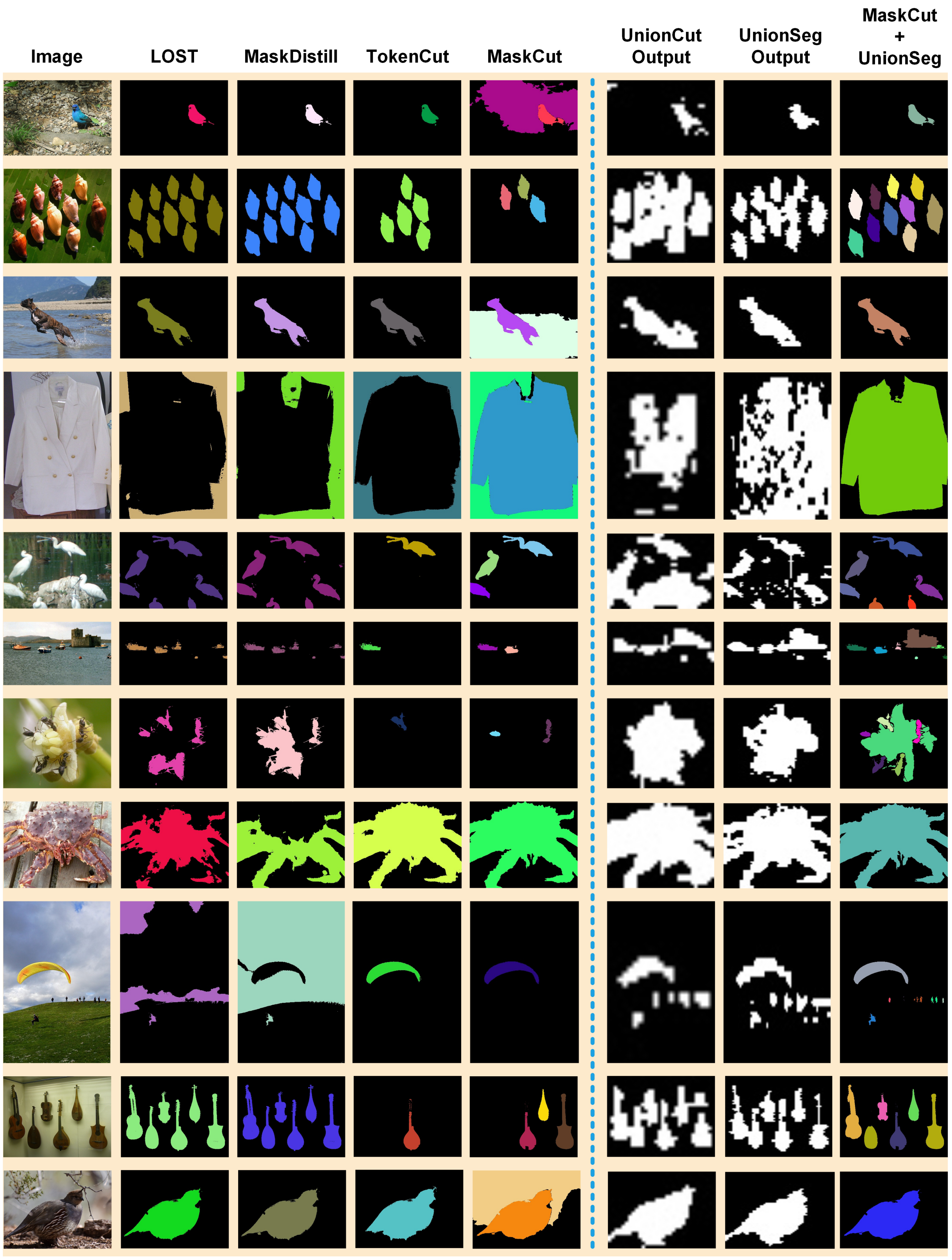


Figure 12. **Visualization of results given by different UOD algorithms.** Masks of the same colour are the result of a one-time discovery.