ProMerge: Prompt and Merge for Unsupervised Instance Segmentation

Dylan Li[1,*] and Gyungin Shin[2,*✉]

[1] Meta Reality Labs
[2] Visual Geometry Group, University of Oxford
https://www.robots.ox.ac.uk/~vgg/research/promerge

Abstract. Unsupervised instance segmentation aims to segment distinct object instances in an image without relying on human-labeled data. This field has recently seen significant advancements, partly due to the strong local correspondences afforded by rich visual feature representations from self-supervised models (e.g., DINO). Recent state-of-the-art approaches use self-supervised features to represent images as graphs and solve a generalized eigenvalue system (i.e., normalized-cut) to generate foreground masks. While effective, this strategy is limited by its attendant computational demands, leading to slow inference speeds. In this paper, we propose Prompt and Merge (ProMerge), which leverages self-supervised visual features to obtain initial groupings of patches and applies a strategic merging to these segments, aided by a sophisticated background-based mask pruning technique. ProMerge not only yields competitive results but also offers a significant reduction in inference time compared to state-of-the-art normalized-cut-based approaches. Furthermore, when training an object detector using our mask predictions as pseudo-labels, the resulting detector surpasses the current leading unsupervised model on various challenging instance segmentation benchmarks.

Keywords: Unsupervised Instance Segmentation · Prompt and Merge

1 Introduction

Instance segmentation identifies and delineates each distinct object within an image, providing both its class and precise pixel-wise location. This capability is crucial for a wide range of applications, from autonomous driving systems [8] that must navigate complex environments to medical imaging technologies that require accurate tumor segmentation [1, 19]. However, the cost of manually annotating dense masks for training data is prohibitively high, especially for domains such as medical imaging that require deep expertise.

To overcome the challenges with dense annotations, multiple endeavors have attempted to tackle category-agnostic instance segmentation in an unsupervised

*Equal contribution
✉Correspondence to: gyungin@robots.ox.ac.uk

Fig. 1: Qualitative examples of ProMerge, a simple yet effective *training-free* approach for unsupervised instance segmentation. Despite its simplicity, ProMerge demonstrates strong segmentation performance.

manner,[3] among which normalized-cut-based approaches [33, 34] have recently shown promise. These methods partition a graph representation of an image encoded in feature space into similar parts and solve a generalized eigenvalue system, specifically through spectral clustering with normalized-cut [27]. Notably, the recently introduced MaskCut method [33], which iteratively employs the TokenCut algorithm [34] on a single image multiple times to generate numerous instance masks, demonstrates state-of-the-art performance. However, repeatedly solving this generalized eigenvalue problem incurs significant computational demands that delay the per-image inference time. Furthermore, its reliance on a fixed criterion to determine the number of segmentation masks per image (with MaskCut using three) may not capture the complete taxonomy of objects in dense, intricate scenes.

In our paper, we propose a simple yet effective framework called ProMerge, which sidesteps the aforementioned limitations. We start with generating initial masks of locally grouped patches by point-prompting self-supervised visual features (e.g., DINO [5]). The initial masks, generated through computing the affinity between individual local patches and global patches, constitute a large set of mask proposals covering different parts of a given image. Following this mask generation process, we iteratively merge these local masks based both on their pixel overlap and their similarity in feature space. The effectiveness of ProMerge is demonstrated through its faster inference speed (about 3.6 times) and competitive performance compared to existing training-free[4] unsupervised methods on six benchmarks, including the densely annotated LVIS [14] and SA-1B [20] datasets. Moreover, by training an object detector (i.e., Cascade Mask R-CNN [3]) with the mask predictions by ProMerge as pseudo-labels, we show that our framework outperforms the current leading unsupervised model (i.e., CutLER [33]).

Our contributions are three-fold: (i) We propose the ProMerge framework, composed of an initial mask generation step using point-prompted self-supervised

[3] In this paper, we consider the *class-agnostic* setting following the prior works [33, 34] on unsupervised instance segmentation.

[4] Here, training-free is defined as not requiring training for segmentation.

visual features, an iterative mask merging that considers similarities in pixel and feature spaces, and a sophisticated mask pruning strategy; (ii) We compare the competitive performance of our approach to the popular normalized-cut-based methods on six standard instance segmentation benchmarks, including COCO2017 [22], COCO-20K [31], LVIS [14], KITTI [13], and subsets of Objects365 [26] and SA-1B [20]; (iii) When training a class-agnostic object detector using the predictions by ProMerge as pseudo-labels, we show the resulting detector outperforms the leading unsupervised detector on the above datasets.

2 Related work

Our work is connected to three themes in the literature, including self-supervised visual representation learning, unsupervised single object detection/segmentation and unsupervised instance segmentation.

2.1 Self-supervised visual representation learning

Self-supervised learning in computer vision has advanced significantly by adopting the principle of learning from the intrinsic structure of data, drawing inspiration from how language models such as GPT [2] and BERT [11] achieve semantic understanding from text. One strategy in self-supervised learning involves leveraging pretext tasks, such as those employed by Masked Autoencoders [15], wherein models learn by predicting the obscured parts of an image. Another set of strategies, embodied by SwAV [4], MoCo [6, 7, 16], and DINO [5, 9, 24], uses data augmentations to generate varied perspectives of images and aligns feature representations of these perspectives. Among these self-supervised paradigms for training encoders, DINO in particular encodes detailed segmentation information, a capability diminished in models trained with supervised labels [5, 28]. In this work, by leveraging DINO's inherent grouping ability, we propose a simple yet effective approach to instance segmentation without explicit labels.

2.2 Unsupervised single object detection and segmentation

Unsupervised object detection and segmentation aim to localize a single, dominant object in an image in an unsupervised manner with a bounding box or a segmentation mask, respectively.

LOST [29] extracts features from a self-supervised network and isolates the foreground by first identifying the seed patch with the lowest count of positive correlations with other patches. A seed expansion strategy is then used to include additional patches that correlate positively with the original seed patches.

Another line of work uses normalized-cut-based methods [23, 28, 34] to distinguish the foreground from the background. This class of methods uses the eigendecomposition of the Laplacian matrix derived from a feature affinity matrix constructed with self-supervised features. The resulting eigenvectors, processed with traditional clustering or thresholding methods, can be translated into meaningful segmentations in pixel space.

This process facilitates the identification of coherent regions, enabling the separation of primary image elements from the background. While these prior works serve as foundational methods for unsupervised multi-object localization and segmentation, they primarily focus on segmenting or identifying the location of a *single* salient object, limiting their generalizability and performance on multi-object localization tasks.

2.3 Unsupervised instance segmentation

Unsupervised instance segmentation aims to identify *multiple* objects in an image without human labels, introducing a more complex challenge than the aforementioned single object detection and segmentation. An early attempt [30] generates a single salient mask per image, and trains the Mask R-CNN detector [17] using the generated masks as pseudo-labels. However, the single mask generation approach does not provide sufficient mask instances per image, resulting in suboptimal detection performance.

Recent approaches [32, 33], on the other hand, propose methods for generating multiple instance masks per image, that are used to train a general object detector. MaskCut [33] has demonstrated promising outcomes by iteratively applying a normalized-cut-based single object segmentation technique (i.e., TokenCut [34]) to images and training a robust object detector with pseudo-labels generated through the MaskCut algorithm. Despite its effectiveness, MaskCut uses repeated eigenvalue system resolutions for the normalized cut, incurring significant computational demands. Additionally, MaskCut limits itself to three segmentations per image. In contrast, ProMerge eschews computationally intensive eigenvalue calculations in favor of using raw feature affinities and does not impose a restriction on the number of segmentations per image. By doing so, it offers a more computationally efficient alternative for the instance segmentation task while achieving higher precision and recall.

3 Method

In this section, we first describe the problem scenario (Sec. 3.1) and introduce ProMerge, a simple yet effective prompt-and-merge method to tackle unsupervised instance segmentation (Sec. 3.2). We then describe a background-based mask pruning strategy to extract foreground masks from a set of prompted masks that increases the effectiveness of prompting and merging (Sec. 3.3). We finally describe a pseudo-label training scheme in which an object detector is trained with pseudo-labels generated by ProMerge (Sec. 3.4).

3.1 Problem formulation

We address the challenging task of instance segmentation in an unsupervised manner. Given an image $\mathbf{I} \in \mathbb{R}^{3\times H\times W}$, where H and W denote the height and width of the image, we aim to produce a set of N instance binary masks

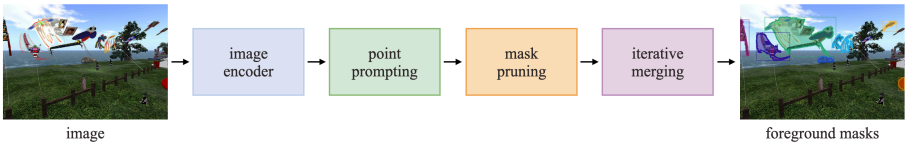


Fig. 2: An overview of ProMerge. Given an input image, we obtain initial mask proposals by prompting visual features from an image encoder using a 2D point grid. Then, the noisy proposals are filtered through the proposed background-based mask pruning. The final predictions are made by iteratively merging the remaining foreground masks.

$\mathcal{M} = \{\mathbf{M}_l \in \{0,1\}^{H \times W} | l = 1, ..., N\}$ without relying on any human-labeled data.

3.2 ProMerge: Prompt and Merge

Point-prompting visual features. Our approach begins with generating preliminary mask proposals. We use visual features, denoted as $\mathbf{F} = \{\mathbf{f}_{ij} \in \mathbb{R}^c | i = 1, ..., h, \ j = 1, ..., w\}$, that are obtained by feeding an image into an image encoder (e.g., patch tokens for Vision Transformers (ViT) [12]). Here, c, h, and w represent the channel, height, and width dimensions of the features. To generate initial masks from the visual features (i.e., patch tokens), we consider the technique of point-prompting, wherein a 2D grid of K equally spaced patch tokens is selected as seeds for mask generation. This subset of the selected tokens $\mathcal{P} = \{\mathbf{p}_l \in \mathbb{R}^c | l = 1, ..., K\}$, which we call the set of *prompt tokens*, is individually compared with all of the patch tokens in the image. By comparing each prompt token in a one-to-all manner via a similarity measure, we generate an affinity matrix $\mathbf{A}_l \in [-1, 1]^{h \times w}$ for each prompt token $\mathbf{p}_l$. In this paper, we use *key* features from the last attention layer of a self-supervised image encoder (i.e., DINO) as patch tokens [33] and compute the cosine similarity between them. That is,

$$\mathbf{A}_l = (A_{l;ij}) = \frac{\mathbf{p}_l \cdot \mathbf{f}_{ij}}{||\mathbf{p}_l||_2 ||\mathbf{f}_{ij}||_2} \tag{1}$$

where $||.||_2$ denotes L2 norm. We then apply a bipartition threshold, τ_b, to the affinity matrix to obtain a binary mask $\mathbf{M}_l$.

Merging prompted masks. Given the prompted masks above, we consider an iterative clustering method, wherein masks, sorted by area in descending order, are sequentially merged with a set of larger masks processed at the previous iterations. The processed masks serve as bases for merging smaller masks that are introduced in later iterations.

In comparing a new, smaller mask with a previously processed, larger mask, we consider two straightforward conditions. First, we use the Intersection-over-Area (IoA) metric to determine if the smaller mask should be merged with the larger mask. If the ratio of the intersection area between two masks to the

area of the smaller mask exceeds a certain threshold, denoted as $\tau_{\text{IoA}}^{\text{merge}}$, the smaller mask is combined with the larger. We choose IoA over the more conventional Intersection-over-Union (IoU) due to IoU's diminished effectiveness in cases where a large mask merges with a significantly smaller one, such as when combining an object's main body with an appendage. Note that the IoA criterion alone is insufficient, as it prevents merging two masks unless they exhibit substantial overlap in pixel space. To allow for merging semantically similar masks with a small overlap, we consider the second condition based on feature similarity. Given a pair of masks, we compute the cosine similarity between their average patch embeddings and merge them if the feature similarity is over a threshold τ_f^{merge}. If a new mask meets at least one of the merging conditions with more than one previously processed mask, the new mask and all compatible masks are merged together. Conversely, if it does not satisfy a merge condition with any of the previously processed masks, it remains unchanged. This mask will then be compared with subsequent masks in later iterations for potential merging.

In summary, through our merge algorithm, we start with disparate, potentially overlapping masks, and end up with semantically-related mask groupings.

3.3 Background-based mask pruning

While the prompt-and-merge framework thus far is intuitive, we observe that it suffers from poor performance in isolation, due to the noisy background masks among the prompted masks. We introduce a mask pruning strategy based on background prediction between the prompting and merging steps. We rely on a two-step process that (i) groups the initial K masks into the foreground or background, and aggregates the background masks via pixel-wise voting to produce a single, fine background mask for the image and (ii) uses the predicted background mask to filter out noisy foreground masks from the initial mask proposals. Each step in the mask pruning strategy is detailed below.

Background aggregation. Recall that after prompting the visual features with K equally spaced points in a 2D grid, we obtain K binary masks. We then classify each of the masks as either a foreground or background candidate. We employ a simple heuristic: a background mask is likely to contain a majority of pixels along the edge for at least two of the edges of the image. Specifically, we consider a mask as background if more than one of its sides contains a number of positive pixels that exceeds half the length of that side. Then, we create a single representative background mask for the image by applying a pixel-wise voting scheme to the background candidates. Formally, given the set of background candidate masks $\mathcal{B} = \{\mathbf{M}_l^{\text{bg}}\}$, a pixel value at (i, j) of the aggregated background mask $\tilde{M}_{ij}^{\text{bg}}$ is determined with:

$$\tilde{M}_{ij}^{\text{bg}} = \left[\frac{\sum_{l=1}^{|\mathcal{B}|} M_{l;ij}^{\text{bg}}}{|\mathcal{B}|} > 0.5 \right] \tag{2}$$

where the [.] operator is the indicator function, which returns 1 if the condition within the operator is satisfied, and 0 otherwise. The condition in the operator

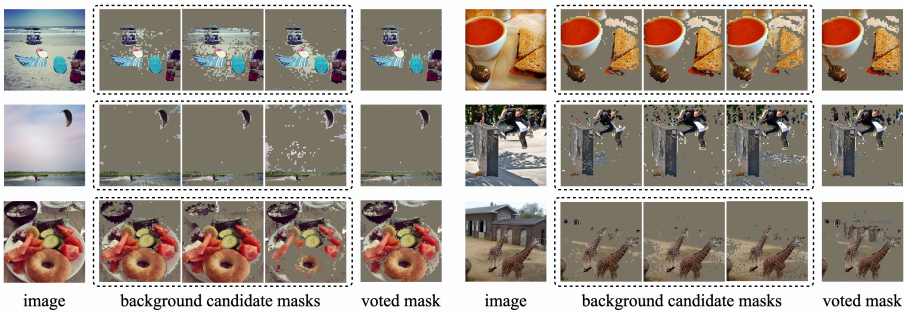


Fig. 3: Qualitative examples of the pixel-wise voting. For each case, an input image, background candidate masks (only three masks are shown for visual purposes), and the voted mask are visualized. The voted background mask, $\tilde{\mathbf{M}}^{\text{bg}}$ effectively filters out the background, leaving only foreground regions despite the noisy candidate masks.

checks whether over the half of the background candidate masks have a value of 1 at (i, j). As such, we obtain a single representative background mask per image, represented by $\tilde{\mathbf{M}}^{\text{bg}} \in \{0, 1\}^{h \times w}$. Some visual examples are shown in Fig. 3.

Foreground filtering. With the background mask $\tilde{\mathbf{M}}^{\text{bg}}$, we exclude prompted masks that are more likely to belong to the background before the merge process. For this filtering step, we consider three separate approaches: (i) intersection-based (ii) similarity-based filtering and (iii) the proposed Cascade filtering.

For intersection-based filtering, we use a simple prior that any foreground masks considered for the following merge process should not significantly intersect with the voted background $\tilde{\mathbf{M}}^{\text{bg}}$. Similarly to Sec. 3.2, we use IoA, as we want the metric to be invariant to the size disparity between a candidate foreground mask and $\tilde{\mathbf{M}}^{\text{bg}}$. If the intersection of a foreground mask and $\tilde{\mathbf{M}}^{\text{bg}}$ divided by the area of the mask is greater than $\tau^{\text{bg}}_{\text{IoA}}$, we regard the mask as belonging to the background and exclude it in the merging process.

For similarity-based filtering, we prune masks with a high similarity with the background mask in feature space. We again compute the mean patch embeddings for the candidate foreground mask and $\tilde{\mathbf{M}}^{\text{bg}}$, before calculating their normalized cosine similarity. If the similarity value is over a threshold, we exclude the mask from the merging process.

In the proposed Cascade filtering approach (shown in Fig. 4), we initially sort the prompted masks in ascending order based on their area. We then proceed through these masks sequentially, maintaining a cumulative ledger of pixels already incorporated into previous masks. At each step, we identify pixels in the current mask that have not yet been considered in a prior mask. We then calculate the IoA between these 'new' pixels and $\tilde{\mathbf{M}}^{\text{bg}}$. We also compute the feature similarity between the current mask and $\tilde{\mathbf{M}}^{\text{bg}}$ in a manner similar to the aforementioned similarity-based filtering. If either of these measures with $\tilde{\mathbf{M}}^{\text{bg}}$

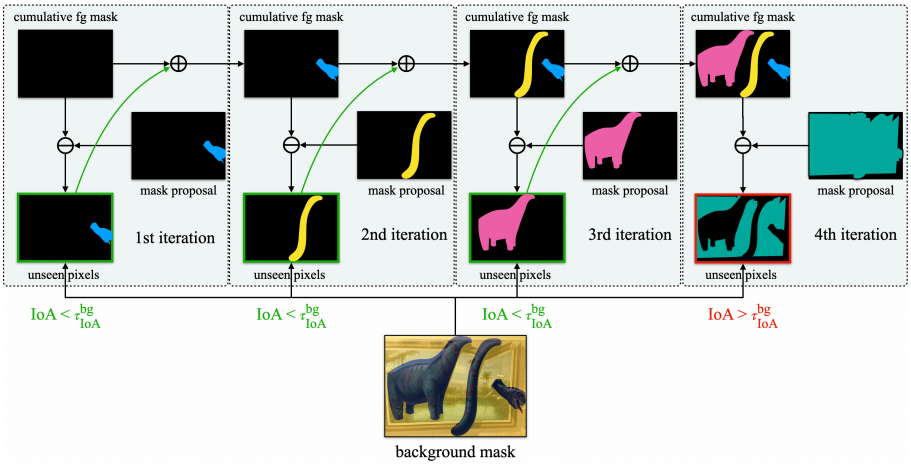


Fig. 4: An illustration of the proposed Cascade mask filtering process. For each iteration of the proposed method, we evaluate the newly proposed mask by focusing on the pixels that have not yet been covered by the cumulative foreground mask, which is an aggregation of pixels from mask proposals in preceding iterations. If these previously unseen pixels demonstrate a significant overlap with the background mask, quantified by the Intersection-over-Area (IoA) metric, the mask proposal for that iteration is subsequently disregarded. An example of this can be observed in the fourth iteration (rightmost), in which the mask proposal is eliminated due to its high IoA with the background mask. Note that in the figure, the feature similarity condition is not shown for visual clarity. See the text for details.

exceeds their respective thresholds, the mask is excluded from the subsequent merge process.

3.4 ProMerge+: Training an object detector with pseudo-labels from ProMerge

Following [33], we train an object detector on ProMerge predictions generated from inference on images in a large-scale image dataset (i.e., ImageNet2012 [10]). The purpose of this pseudo-label training is two-fold: firstly, to obtain an object detector with better performance by learning from the noisy pseudo-labels; and secondly, to assess the detector's ability to generalize across different data distributions by training on images from one dataset (e.g., ImageNet2012) and evaluating on another (e.g., SA-1B [20]). The trained detector, ProMerge+, surpasses performance and zero-shot transfer capabilities of the current leading model.

4 Experiments

In this section, we first provide implementation details (Sec. 4.1) and compare our method with the state-of-the-art-methods (Sec. 4.2). Then, we provide extensive ablation study to analyze our approach (Sec. 4.3).

4.1 Implementation details

Our implementation is based on the PyTorch [25] and Detectron2 [35] libraries, and A100 GPUs are used for our experiments unless otherwise stated.

Datasets. We evaluate our ProMerge on six benchmarks including COCO2017 [22], COCO-20K [31], LVIS [14], KITTI [13], subsets of Objects365 [26] and SA-1B [20] (44K and 11K images, respectively). Among these, SA-1B is the most challenging benchmark due to the densely-annotated fine-grained masks, with an average of 101 segmentation masks per image. To train ProMerge+, we use unlabeled ImageNet2012 training images [10] (1.2M images) and evaluate the resulting model on the six benchmarks in a zero-shot manner (i.e., the model is not trained with images sharing the same data distribution as evaluation data). For the ablation study, we use COCO2017 following [33].

Evaluation metrics. We evaluate our methods based on average precision (AP) and recall (AR), the standard metrics for the instance segmentation task.

Inference of ProMerge. We follow the previous work [33] for our inference setting. Specifically, we use DINO [5] with the ViT-B/8 architecture [12] as an image encoder and input images are resized to 480×480 pixels before being fed into the encoder. We apply Conditional Random Field (CRF) [21] to an output mask for post-processing. Additionally, we split connected components of initial masks before merging, which we find beneficial in terms of both AP and AR (shown in Sec. 4.3).

Pseudo-label training. When we train an object detector with pseudo-masks generated by ProMerge, we use the Cascade Mask R-CNN model [3] with the ResNet50 backbone [18] initialized with DINO features [5]. We train the detector for 160K iterations using the SGD optimizer with a batch size of 16, a momentum of 0.9, and a learning rate of 0.005, which is decreased by a factor of 5 during training.

4.2 Main results

In this section, we compare our methods to the state-of-the-art methods with both standard evaluation metrics and inference speed.

Comparison to state-of-the-art methods. We first compare ProMerge to *training-free* methods including TokenCut [34] and MaskCut [33] algorithms. In Tab. 1 (top), ProMerge demonstrates superior performance in both average precision (AP) and average recall (AR_{100}) compared to the existing methods. The overall higher recall of ProMerge is attributed to its flexibility in not requiring a

method	COCO2017 AP^{mk}	COCO2017 AR^{mk}_{100}	COCO-20K AP^{mk}	COCO-20K AR^{mk}_{100}	LVIS AP^{mk}	LVIS AR^{mk}_{100}	KITTI AP^{mk}	KITTI AR^{mk}_{100}	Objects365† AP^{bb}	Objects365† AR^{bb}_{100}	SA-1B AP^{mk}	SA-1B AR^{mk}_{100}	Average AP	Average AR_{100}
Training-free methods														
TokenCut [34]	2.0	4.4	2.7	4.6	0.9	1.8	**0.3**	1.5	1.1	2.1	1.0	0.3	1.3	2.5
MaskCut [33]	2.2	6.9	**3.0**	6.7	0.9	2.6	0.2	**2.2**	1.7	4.0	0.8	0.6	1.5	3.5
ProMerge	**2.4**	**7.5**	**3.0**	**7.4**	**1.3**	**3.3**	**0.3**	1.9	**2.2**	**6.0**	**1.2**	**0.8**	**1.7**	**4.5**
Models trained with pseudo-labels														
CutLER‡ [33]	8.7	24.9	8.9	25.1	3.4	16.6	3.9	23.3	11.5	34.3	5.5	13.5	7.0	22.9
ProMerge+	**8.9**	**25.1**	**9.0**	**25.3**	**4.0**	**17.7**	**5.4**	**25.7**	**12.2**	**35.8**	**7.8**	**16.3**	**7.9**	**24.3**

Table 1: Comparison between training-free methods (top) and models trained with pseudo-labels (bottom). CutLER and ProMerge+ are trained with pseudo-labels generated by MaskCut and ProMerge, respectively. †Only ground-truth bbox annotations available. ‡Re-implemented with a single round of training for a fair comparison.

method	FPS
TokenCut [34]	0.34
MaskCut [33]	0.15
ProMerge	**0.54**

Table 2: Speed comparison. Our method is approximately 3.6 times faster in FPS compared to MaskCut.

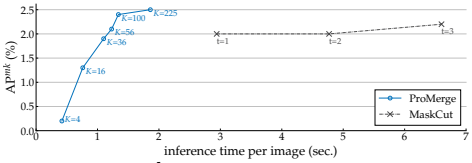


Fig. 5: AP^{mk} vs inference time on COCO2017. K denotes the number of prompt tokens. MaskCut uses t=3 by default in the original paper.

predetermined number of masks per image. Qualitative results of each method can be found in the supplementary.

We next compare ProMerge+, a class-agnostic object detector trained with the predictions of ProMerge on ImageNet as pseudo-labels, to the state-of-the-art model (i.e., CutLER [33]). As can be seen in Tab. 1 (bottom), ProMerge+ outperforms CutLER [33] by 0.9 and 1.4 in AP and AR_{100}, respectively, on average. Note that we use the same detector architecture (i.e., Cascade Mask R-CNN [3]) and the training recipe as CutLER. The sole difference is that CutLER is trained with predictions from MaskCut and ProMerge+ is trained with ones from ProMerge. These results demonstrate the effectiveness of using our higher quality pseudo-labels for training an object detector.

Speed comparison. Unlike the state-of-the-art training-free methods that depend on solving the computationally intensive normalized-cut algorithm, our method avoids this complexity, enabling faster inference. To demonstrate this, we compare the inference speed of our method with that of normalized-cut-based methods in terms of Frames Per Second (FPS). In detail, we measure the timing for each of TokenCut, MaskCut, and ProMerge across 200 images, using the first 100 images as a warm-up phase to ensure accurate measurements.[5] As shown in Tab. 2, ProMerge runs at 0.54 FPS, which is about 1.6 and 3.6 times faster than

[5]We use a single RTX 3080 GPU and a 12th Gen Intel(R) Core(TM) i7-12700K chipset for this experiment.

Prompting	Merging	Mask Pruning	CC Splitting	AP^{mk}_{50}	AP^{mk}	AR^{mk}_{100}
✓	✗	✗	✗	0.8	0.5	1.6
✓	✓	✗	✗	0.6	0.4	1.0
✓	✓	✓	✗	4.4	2.1	4.7
✓	✓	✓	✓	**5.6**	**2.4**	**7.5**

Table 3: Effect of individual components. A naive prompting and merging approach suffers poor performance, while applying the proposed background-based mask pruning (Mask Pruning) allows for a notable increase in performance, which is further enhanced by connected component splitting (CC Splitting). Default settings are marked in gray.

TokenCut and MaskCut, respectively. In addition, we compare the speed differential between ProMerge and MaskCut under various inference configurations. For MaskCut, we adjust the number of repetitions (t) for solving the eigenvalue problem, which significantly impacts its runtime. For our method, we vary the number of prompt tokens (K). As can be seen in Fig. 5, ProMerge provides a more advantageous trade-off between inference speed and performance in AP^{mk} compared to MaskCut. It achieves higher AP^{mk} values when K exceeds 100, while also being significantly faster—approximately 4.9 times faster at $K = 100$.

4.3 Ablation study

Here, we conduct an extensive ablation study to analyze the effect of each component in ProMerge. Specifically, we explore the effects of pixel-wise voting for background aggregation, foreground filtering methods, different merging conditions, and varied hyperparameter choices.

Effect of each component. We identify major components that affect the performance of our approach: (i) prompting and merging; (ii) background-based mask pruning (denoted as Mask Pruning); and (iii) connected component splitting (denoted as CC Splitting). In Tab. 3, we show the influence of each component. Notably, a naive approach that relies solely on prompting and merging suffers from poor performance, with 0.4 AP^{mk} and 1.0 AR^{mk}_{100}, which are worse than a prompting-only approach. However, adding background-based mask pruning greatly boosts both AP^{mk} and AR^{mk}_{100} by 1.7% and 3.7%. CC-splitting further increases AP^{mk} by 0.3% and AR^{mk}_{100} by 2.8%. These results demonstrate that though the core of prompt-and-merge is straightforward, the competitive performance of our approach is facilitated by incorporating sophisticated components such as background mask pruning.

Effect of pixel-wise voting. In the background-based mask pruning process, we use a pixel-wise voting strategy to aggregate background candidate masks to produce a single, representative background mask for a given image. We consider the case where we substitute voting with a simpler non-voting mechanism, and compare it with the voting based mechanism in ProMerge. In the non-voting experiment, we sum up all background candidate masks. If a pixel at location (i, j) has at least one mask with a value of one (i.e., positive pixel), it is regarded

voting	AP^{bb}_{50}	AP^{bb}	AR^{bb}_{100}	AP^{mk}_{50}	AP^{mk}	AR^{mk}_{100}
✗	4.1	1.6	5.0	3.2	1.2	4.2
✓	**6.0**	**3.0**	**8.6**	**5.6**	**2.4**	**7.5**

(a) Effect of voting

filter. method	AP^{bb}_{50}	AP^{bb}	AR^{bb}_{100}	AP^{mk}_{50}	AP^{mk}	AR^{mk}_{100}
IoA	4.7	2.1	6.8	4.0	1.7	5.8
feat.	5.0	2.3	5.9	4.6	2.0	5.2
Cascade	**6.0**	**3.0**	**8.6**	**5.6**	**2.4**	**7.5**

(b) Impact of filtering criteria

Table 4: Influence of the voting strategy and filtering conditions. (Left) We note that using the proposed pixel-wise voting for obtaining a representative background mask allows for a notable gain in performance. **(Right)** Comparing the Intersection-over-Area-based filtering (denoted as IoA) and the feature similarity-based filtering (denoted as feat.), the former demonstrates higher recall, whereas the latter excels in precision. Our proposed Cascade filtering outperforms both in all evaluated metrics, showcasing its effectiveness.

as a background pixel. As highlighted in Tab. 4a, voting leads to a significant enhancement in both average precision and recall. In the absence of pixel-wise voting, the aggregated background often encompasses not only the actual background but also parts of the foreground in the image, which detracts from the overall performance. Conversely, the application of the voting strategy more accurately isolates the background region.

Effect of foreground filtering method. Given the background mask obtained from the pixel-wise voting above, we filter prompted masks based on the background mask as described in Sec. 3.3. Here, we compare three different methods including (i) IoA-based, (ii) feature-similarity-based, and (iii) our proposed Cascade filtering strategies. For the IoA-based filtering, we discard a mask proposal if it has an IoA with the background mask exceeding 0.5, meaning more than half of the proposal's pixels are part of the background. In the feature-similarity-based approach, we classify a proposal as background if the cosine similarity between the normalized mean embeddings of the candidate foreground mask and the background exceeds 0, implying that the proposal's embedding closely aligns with that of the background.

From Tab. 4b, we make the following key observations. First, IoA-based filtering, demonstrates higher recall but lags in precision compared with similarity-based filtering. This discrepancy can be attributed to the latter's tendency to filter out more masks, irrespective of their degree of area overlap with the background, which in turn affects recall. Second, our proposed Cascade filtering emerges as the superior method. Its effectiveness stems from retaining smaller object masks while filtering larger masks that might incorrectly merge these smaller ones (in the following merge step). In other words, as Cascade filtering focuses specifically on the new, unseen pixels (and their respective relationships with the background), it lifts precision and recall by preventing the amalgamation of smaller entities into a larger, encompassing mask during mask merging.

Effect of merging conditions. In the merging process, we consider two different conditions for merging: (i) Intersection-over-Area (denoted as IoA) and (ii) feature similarity (denoted as feat.) between two masks. As shown in Tab. 5, we

feat.	IoA	AP^{bb}_{50}	AP^{bb}	AR^{bb}_{100}	AP^{mk}_{50}	AP^{mk}	AR^{mk}_{100}
✗	✓	5.3	2.3	7.9	4.4	1.9	6.9
✓	✗	5.8	2.7	8.4	4.9	2.2	7.2
✓	✓	**6.0**	**3.0**	**8.6**	**5.6**	**2.4**	**7.5**

Table 5: Effect of merging conditions. Considering both feature similarity (denoted as feat.) and intersection-over-area (denoted as IoA) for merging a pair of masks yields further gain.

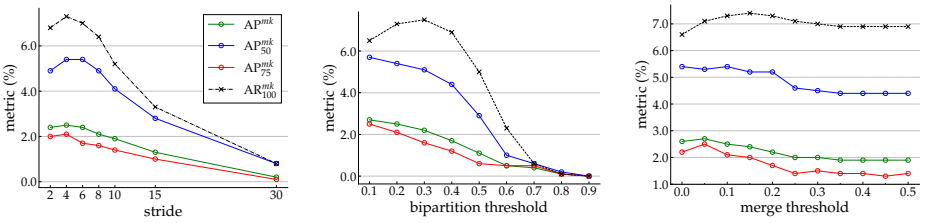
observe that the feature similarity condition plays a more significant role than the IoA condition when used separately. However, allowing both conditions leads to the best performance, indicating that they are complementary conditions.

Hyperparameter analysis. Lastly, we conduct experiments to explore the impact of various hyperparameters, including grid size, bipartition threshold for obtaining initial masks (τ_b), and feature similarity threshold for merging (τ_f^{merge}). We refer to the grid size parameter as 'stride', where the stride is inversely proportional to the grid size. For instance, with a stride of 2 and a spatial dimension of 60×60 of DINO feature embeddings, the grid size is reduced to 30×30, resulting in 900 initial prompted masks (i.e., $K = 900$). Conversely, a stride of 30 results in a grid size of 2×2, yielding only 4 initial prompted masks (i.e., $K = 4$).

In Fig. 6a, we present the performance of ProMerge using strides of {2, 4, 6, 8, 10, 15, 30}. The stride significantly influences performance, with a stride of 4 yielding the best results, while larger strides such as 15 and 30 lead to diminished performance. This decline is attributed to the reduced number of masks generated at higher stride values.

We next examine the performance variation associated with different values of the bipartition threshold τ_b, which is employed to binarize initial soft masks derived from prompting. A lower τ_b results in prompted masks having a larger area, while a higher τ_b leads to smaller mask areas. As illustrated in Fig. 6b, optimal average precision and recall are achieved between 0.1 and 0.3, with the highest precision at $\tau_b = 0.1$ and the highest recall at $\tau_b = 0.3$. Performance in both metrics declines as τ_b increases, suggesting that ProMerge benefits more from larger initial masks. Consequently, we set τ_b to 0.2 for all our experiments.

We also investigate the impact of the feature similarity threshold τ_f^{merge}, which determines whether two masks should be merged. A lower τ_f^{merge} value leads to the merging of mask pairs even with minimal similarity, while a higher value requires a high degree of similarity for merging. Figure 6c demonstrates that our method's performance significantly varies with τ_f^{merge} values between 0.0 and 0.3, but levels off at higher values. This plateau suggests that beyond a τ_f^{merge} of 0.3, few mask pairs meet the merging criterion, thereby minimally impacting the overall performance metrics. Consequently, we opt for a $\tau_f^{\text{merge}} = 0.1$ in our experiments.



(a) Effect of stride **(b)** Effect of τ_b **(c)** Effect of τ_f^{merge}

Fig. 6: Hyperparameter analysis. We note that stride (i.e., the Euclidean distance between two nearest point prompts in a regular 2D grid), bipartition threshold (τ_b) for binarizing prompted masks, and feature similarity threshold (τ_f^{merge}) for merging masks play important roles in our approach. The default setting uses stride=4, τ_b=0.2, and τ_f^{merge}=0.1, respectively. Best viewed in color.

5 Discussion

We observe that a simple approach of prompting and merging masks is competitive with computationally intensive normalized-cut-based approaches [33, 34]. We primarily attribute the success of ProMerge to two properties: (i) unlike in [33, 34], we do not assume a fixed number of mask predictions per image and allow the algorithm to flexibly make predictions, which we find particularly helpful when multiple objects are present in a given image; (ii) we use the proposed sophisticated background-based filtering method, which excludes masks that overlap with the background of an image.

ProMerge shows promise but still has room to improve compared to fully-supervised methods, such as SAM [20]. We believe that the gap partly results from utilizing self-supervised features that are not trained with a pretext task driven by object localization or segmentation. Training and adopting self-supervised features specifically tailored for object localization or segmentation could significantly bridge this gap.

6 Conclusion

Our work introduces Prompt and Merge (ProMerge), a novel method to unsupervised instance segmentation that capitalizes on the strong local correspondences afforded by self-supervised visual features. By iteratively merging initial patch groupings and employing a sophisticated background-based mask pruning technique, ProMerge achieves competitive performance with a significant reduction in inference time compared to state-of-the-art normalized-cut-based methods on standard benchmarks. Moreover, the application of our method in training an object detector with pseudo-labels demonstrates superior performance, surpassing the leading unsupervised segmentation model.

Acknowledgements. This work was performed using resources provided by the Cambridge Service for Data Driven Discovery (CSD3) operated by the University of Cambridge Research Computing Service (www.csd3.cam.ac.uk), provided by Dell EMC and Intel using Tier-2 funding from the Engineering and Physical Sciences Research Council (capital grant EP/T022159/1), and DiRAC funding from the Science and Technology Facilities Council (www.dirac.ac.uk). All authors appreciate the anonymous reviewers for the thoughtful suggestions and Fengting Yang for the kind advice on the rebuttal. DL would like to also thank Eric Tran and Robert Wang for their support while DL conducted his research at Meta. GS would like to thank Zheng Fang for the enormous support.

References

1. Bilic, P., Christ, P., Li, H.B., Vorontsov, E., Ben-Cohen, A., Kaissis, G., Szeskin, A., Jacobs, C., Mamani, G.E.H., Chartrand, G., Lohöfer, F., Holch, J.W., Sommer, W., Hofmann, F., Hostettler, A., Lev-Cohain, N., Drozdzal, M., Amitai, M.M., Vivanti, R., Sosna, J., Ezhov, I., Sekuboyina, A., Navarro, F., Kofler, F., Paetzold, J.C., Shit, S., Hu, X., Lipková, J., Rempfler, M., Piraud, M., Kirschke, J., Wiestler, B., Zhang, Z., Hülsemeyer, C., Beetz, M., Ettlinger, F., Antonelli, M., Bae, W., Bellver, M., Bi, L., Chen, H., Chlebus, G., Dam, E.B., Dou, Q., Fu, C.W., Georgescu, B., i Nieto, X.G., Gruen, F., Han, X., Heng, P.A., Hesser, J., Moltz, J.H., Igel, C., Isensee, F., Jäger, P., Jia, F., Kaluva, K.C., Khened, M., Kim, I., Kim, J.H., Kim, S., Kohl, S., Konopczynski, T., Kori, A., Krishnamurthi, G., Li, F., Li, H., Li, J., Li, X., Lowengrub, J., Ma, J., Maier-Hein, K., Maninis, K.K., Meine, H., Merhof, D., Pai, A., Perslev, M., Petersen, J., Pont-Tuset, J., Qi, J., Qi, X., Rippel, O., Roth, K., Sarasua, I., Schenk, A., Shen, Z., Torres, J., Wachinger, C., Wang, C., Weninger, L., Wu, J., Xu, D., Yang, X., Yu, S.C.H., Yuan, Y., Yue, M., Zhang, L., Cardoso, J., Bakas, S., Braren, R., Heinemann, V., Pal, C., Tang, A., Kadoury, S., Soler, L., van Ginneken, B., Greenspan, H., Joskowicz, L., Menze, B.: The liver tumor segmentation benchmark (lits). Medical Image Analysis (2023) 1
2. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language models are few-shot learners. In: NeurIPS (2020) 3
3. Cai, Z., Vasconcelos, N.: Cascade r-cnn: Delving into high quality object detection. In: CVPR (2018) 2, 9, 10
4. Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., Joulin, A.: Unsupervised learning of visual features by contrasting cluster assignments. In: NeurIPS (2020) 3
5. Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging properties in self-supervised vision transformers. In: ICCV (2021) 2, 3, 9
6. Chen, X., Fan, H., Girshick, R., He, K.: Improved baselines with momentum contrastive learning. arXiv:2003.04297 (2020) 3
7. Chen, X., Xie, S., He, K.: An empirical study of training self-supervised vision transformers. In: ICCV (2021) 3

8. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: CVPR (2016) 1
9. Darcet, T., Oquab, M., Mairal, J., Bojanowski, P.: Vision transformers need registers. arXiv:2309.16588 (2023) 3
10. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR (2009) 8, 9
11. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: NAACL (Jun 2019) 3
12. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: ICLR (2021) 5, 9
13. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: CVPR (2012) 3, 9
14. Gupta, A., DollÃąr, P., Girshick, R.: Lvis: A dataset for large vocabulary instance segmentation. In: CVPR (2019) 2, 3, 9
15. He, K., Chen, X., Xie, S., Li, Y., DollÃąr, P., Girshick, R.: Masked autoencoders are scalable vision learners. In: CVPR (2022) 3
16. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. In: CVPR (2020) 3
17. He, K., Gkioxari, G., DollÃąr, P., Girshick, R.: Mask r-cnn. In: ICCV (2017) 4
18. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016) 9
19. He, S., Bao, R., Li, J., Stout, J., Bjornerud, A., Grant, P.E., Yangming, O.: Computer-vision benchmark segment-anything model (sam) in medical images: Accuracy in 12 datasets. arXiv:2304.09324 (2023) 1
20. Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.Y., Dollar, P., Girshick, R.: Segment anything. In: ICCV (2023) 2, 3, 8, 9, 14
21. Krähenbühl, P., Koltun, V.: Efficient inference in fully connected crfs with gaussian edge potentials. In: NeurIPS (2011) 9
22. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: ECCV (2014) 3, 9
23. Melas-Kyriazi, L., Rupprecht, C., Laina, I., Vedaldi, A.: Deep spectral methods: A surprisingly strong baseline for unsupervised semantic segmentation and localization. In: CVPR (2022) 3
24. Oquab, M., Darcet, T., Moutakanni, T., Vo, H.V., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., Howes, R., Huang, P.Y., Xu, H., Sharma, V., Li, S.W., Galuba, W., Rabbat, M., Assran, M., Ballas, N., Synnaeve, G., Misra, I., Jegou, H., Mairal, J., Labatut, P., Joulin, A., Bojanowski, P.: Dinov2: Learning robust visual features without supervision. arXiv:2304.07193 (2023) 3
25. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: NeurIPS (2019) 9
26. Shao, S., Li, Z., Zhang, T., Peng, C., Yu, G., Zhang, X., Li, J., Sun, J.: Objects365: A large-scale, high-quality dataset for object detection. In: ICCV (2019) 3, 9
27. Shi, J., Malik, J.: Normalized cuts and image segmentation. TPAMI (2000) 2

28. Shin, G., Albanie, S., Xie, W.: Unsupervised salient object detection with spectral cluster voting. In: CVPRW (2022) 3
29. Siméoni, O., Puy, G., Vo, H.V., Roburin, S., Gidaris, S., Bursuc, A., Pérez, P., Marlet, R., Ponce, J.: Localizing objects with self-supervised transformers and no labels. In: BMVC (2021) 3
30. Van Gansbeke, W., Vandenhende, S., Van Gool, L.: Discovering object masks with transformers for unsupervised semantic segmentation. arXiv:2206.06363 (2022) 4
31. Vo, H.V., Pérez, P., Ponce, J.: Toward unsupervised, multi-object discovery in large-scale image collections. In: ECCV (2020) 3, 9
32. Wang, X., Yu, Z., De Mello, S., Kautz, J., Anandkumar, A., Shen, C., Alvarez, J.M.: Freesolo: Learning to segment objects without annotations. In: CVPR (2022) 4
33. Wang, X., Girdhar, R., Yu, S.X., Misra, I.: Cut and learn for unsupervised object detection and instance segmentation. In: CVPR (2023) 2, 4, 5, 8, 9, 10, 14
34. Wang, Y., Shen, X., Hu, S.X., Yuan, Y., Crowley, J.L., Vaufreydaz, D.: Self-supervised transformers for unsupervised object discovery using normalized cut. In: CVPR (2022) 2, 3, 4, 9, 10, 14
35. Wu, Y., Kirillov, A., Massa, F., Lo, W.Y., Girshick, R.: Detectron2. https://github.com/facebookresearch/detectron2 (2019) 9