Master's Thesis

Tackling the Challenges of Unsupervised Object Detection

Ajesh Krishnan Kizhakke Menakath

Advisers: Silvio Galesso

Examiner: Prof. Dr. Thomas Brox
Prof. Dr. Matthias Teschner

Albert-Ludwigs-University Freiburg
Faculty of Engineering
Department of Computer Science
Chair of Pattern Recognition and Image Processing

August 23th, 2024

Writing period

23. 02. 2024 – 23. 08. 2024

Examiner

Prof. Dr. Thomas Brox

Second Examiner

Prof. Dr. Matthias Teschner

Advisers

Silvio Galesso

Declaration

I hereby declare, that I am the sole author and composer of my thesis and that no other sources or learning aids, other than those listed, have been used. Furthermore, I declare that I have acknowledged the work of others by providing detailed references of said work.
I hereby also declare, that my Thesis has not been prepared for another examination or assignment, either wholly or excerpts thereof.

Place, Date	Signature

Abstract

This thesis focuses on advancing unsupervised object detection and instance segmentation by building upon and improving the CutLER framework. While CutLER has demonstrated significant potential in unsupervised learning tasks, it exhibits limitations, particularly in handling overlapping instances and dependence on initial masks. This work investigates these limitations in detail.

To address these challenges, we propose three key contributions. First, a detailed study of the effects of overlapping instances is conducted, which leads to the decision to remove images with overlapping instances from the training set. This adjustment resulted in observed improvements in model performance. Second, we verify the hypothesis that object-centricity of images contributes to better instance discrimination capabilities. Finally, we introduce a refined mask filtration scheme that improves the quality of the pseudo-ground truth masks generated by the MaskCut method. By removing ambiguous and low-confidence masks during training, the proposed approach effectively reduces noise and enhances the precision of mask predictions.

Experiments conducted across diverse datasets, including COCO and PASCAL VOC, validate the effectiveness of the proposed improvements. The results consistently show performance gains in both detection and segmentation tasks, underscoring the importance of addressing overlapping instances and refining mask annotations in unsupervised learning. This thesis not only seeks to enhance the performance of the CutLER framework but also provides valuable insights into the challenges and opportunities inherent in unsupervised object detection and segmentation.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

Instance detection is a fundamental challenge in computer vision with critical applications in fields ranging from autonomous driving to medical imaging. Recent advancements have seen a shift from traditional Convolutional Neural Networks (CNNs) to the transformative capabilities of Transformers, as demonstrated by Dosovitskiy et al. (2020) with their Vision Transformer (ViT) [1]. Subsequently, the concept of utilizing deep ViT features as dense visual descriptors was introduced by Amir et al. (2021) [2], highlighting the strong semantic information these descriptors provide about instances within an image. This shift highlights a growing interest in leveraging deep, self-supervised features for dense visual representation.

In this context, the current state-of-the-art method for unsupervised instance segmentation, CutLER [3], utilizes DINO features [4] to identify instances within images. Despite its advancements, CutLER faces challenges with overlapping instances and complex background patterns. Addressing these issues not only aims to improve the practical performance of unsupervised methods but also contributes to understanding the efficacy of self-supervised features and the role of inductive biases in computer vision. These methods are crucial not only for their direct applications but also as pre-training techniques for downstream tasks, enhancing models trained with limited labeled data.

This thesis seeks to investigate and overcome these limitations by analyzing the CutLER baseline and proposing possible enhancements to unsupervised object detection and segmentation. By improving these methods, we aim to advance the state of unsupervised learning, providing deeper insights into feature representations and their impact on vision tasks.

1.2 Overview of CutLER

In the field of unsupervised object detection and instance segmentation, the recent work by Xudong Wang et al. introduces Cut-and-LEaRn (CutLER) [3], a novel

Figure 1: Maskcut and CutLER outputs. Figure illustrates the outputs of MaskCut and CutLER with N=3 (Number of masks generated).

approach that significantly advances the state-of-the-art. CutLER leverages the capabilities of self-supervised models to identify objects without human supervision, and it enhances this capability to train a high-performance localization model without any labeled data. The methodology begins with the MaskCut approach (inspired from [5]), which generates coarse masks for multiple objects within an image. Subsequently, a detector is trained on these masks using a robust loss function. Performance is further improved through a self-training process where the model is iteratively trained on its own predictions. This approach not only simplifies the training process but also proves to be compatible with various detection architectures and capable of detecting multiple objects simultaneously. Figure 1 shows the original image, masks generated by Maskcut for N=3, where N represents maximum number of masks generated per image (like in the original paper, we use N=3 in all of our experiments) and the final CutLER output.

The effectiveness of CutLER is demonstrated through extensive evaluations across diverse image domains, including video frames, paintings, sketches, and complex scenes. Notably, CutLER, utilizing a ResNet50 [6] backbone, achieves a substantial performance increase, more than doubling the detection accuracy on 10 out of 11 benchmarks compared to the previous state-of-the-art method, FreeSOLO [7], which uses a ResNet101 [6] backbone. Specifically, CutLER improves the average precision by over 2.7 times across these benchmarks. This demonstrates CutLER's potential not only as a zero-shot unsupervised detector but also as an efficient low-shot detector,

marking a significant step forward in unsupervised object detection and instance segmentation.

1.3 Contribution and Key Insights

This study focuses on the shortcomings of CutLER and the methods to minimize them. Our main contributions are:

1. **Analysis on the Influence of Overlapping Instances**: We compare the performance of the model when trained with and without overlapping instances and conclude that training without overlapping instances results in better instance discrimination.

2. **Exploring the Strength of Object-Centric Prior**: We compare the performance of the model when trained only with object-centric images against the baseline trained with all images.

3. **Proposing New Mask Filtration Method**: On top of self-training, we refine MaskCut masks using our modified mask filtration method to remove inaccurate masks and retraining to yield better performance.

1.4 Outline

- **Chapter 2:** Explores recent advances in self-supervised feature learning and developments in unsupervised object detection and semantic segmentation along with the foundational concepts of Vision Transformers and attention mechanisms, with a detailed introduction to DINO features. Also covers the CutLER training pipeline and the method for mask refinement.

- **Chapter 3:** Provides a detailed explanation of the limitations of CutLER, with descriptive examples of the proposed mask filtration method and the baseline approach. Also includes comprehensive background information for experimenting with the impact of overlapping instances and object-centric priors.

- **Chapter 4:** Details evaluation of the baseline and proposed methods across various datasets, focusing on instance detection and segmentation tasks. It includes a comparison of performance metrics, such as AP and AP50, highlighting improvements and differences between the methods. Additionally, it

examines the impact of overlapping instances and discusses the effectiveness of the proposed mask refinement approach.

2 Background and Related Works

2.1 Vision Transformer

Vision Transformer (ViT), introduced by Dosovitskiy et al. [1], provides an alternative to Convolutional Neural Networks (CNNs) for image recognition, leveraging the transformer architecture to capture long-range dependencies and complex patterns that can complement and extend the capabilities of CNN-based models. The model applies Transformer architecture to image recognition tasks by treating image patches as sequences of tokens, akin to words in Natural Language Processing (NLP). The highlight of the paper is reusing the transformer encoder from the revolutionary work of Vaswani et al. [8] and adapting to use on images using patch tokenization and positional encoding.

As CutLER uses the features from a self-supervised ViT to generate masks, it is crucial to understand the basics architecture and working of ViT to get a complete picture of the feature generation process. Figure 2 shows the complete architecture of ViT. We are going to go into the main parts of the architecture for a better understanding of the process.

2.1.1 Patch Tokens and Positional Encoding

As each input is an image, unlike a sequence of words or tokens in [8], the image is divided in fixed size patches (16x16 or 8x8) and each patch is treated as a token and each token is embedded into a fixed-dimensional vector using a learned embedding layer.

$$z^0{}_i = z_i + p_i \tag{1}$$

For each token, instead of using sinusoidal position encodings [8] to retain information about the position of tokens in the sequence, a learnable position embedding is added as shown in the Eq. 1, where $p_i \in \mathbb{R}^D$ is the learnable position embedding for patch i.

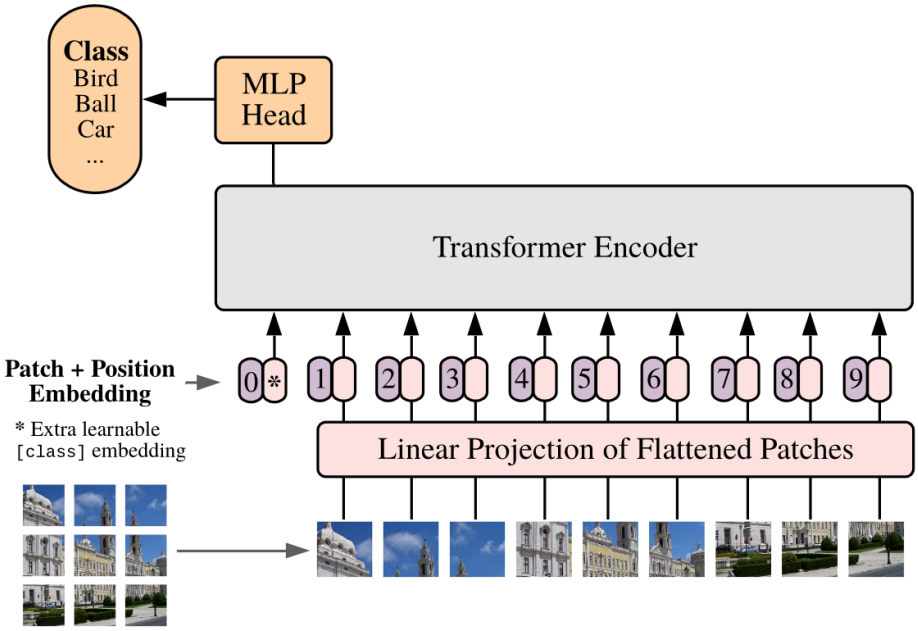

Figure 2: ViT Architecture. ViT architecture from [1].

$$Z = [z_{class}; {z^0}_1; {z^0}_2; ...; {z^0}_N] \tag{2}$$

Apart from [8], ViT [1] introduces a special classification token z_{class} which is prepended to the sequence of patch embeddings. This token aggregates information from all patches and is used for the final classification task. The final encoding looks like Eq. 2.

2.1.2 Transformer Encoder

The sequence of patch embeddings, augmented with positional information, is processed by the Transformer encoder. The encoder consists of multiple layers, each comprising Multi-Head Self-Attention (MSA) and Multi-Layer Perceptrons (MLPs), with Layer Normalization (LN) and residual connections. A weighted average [9] of individual attention outputs constitutes the final output. Figure 3 illustrates the architecture of the transformer encoder. We briefly look into each part.

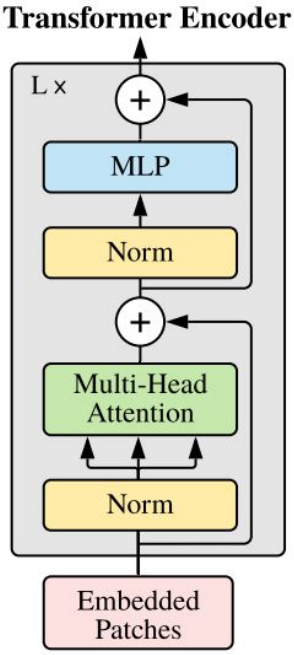

Figure 3: Transformer Encoder Architecture. Illustration of the Transformer encoder architecture in ViT [1].

Multi-Head Self-Attention (MSA)

Self-attention allows the model to weigh the importance of different patches relative to each other.

$$\text{Queries } Q = zW_i^Q \tag{3a}$$

$$\text{Keys } K = zW_i^K \tag{3b}$$

$$\text{Values } V = zW_i^V \tag{3c}$$

Given that d_k is the dimensionality of the key, query, and value vectors and $W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{D \times d_k}$ are learnable weight matrices, query, key, and value are computed as given in Eq. 3.

For each attention head i,

$$head_i = \text{Attention } (Q_i, K_i, V_i) = \text{softmax } \left(\frac{Q_i K_i^T}{\sqrt{d_k}} \right) V_i \tag{4}$$

The outputs from all heads are concatenated and linearly transformed. Given

$W^O \in \mathbb{R}^{h-d_k \times D}$:

$$MSA(z) = \text{Concat } (head_i, head_2..., head_h)W^O \tag{5}$$

Layer Normalization and Residual Connections

Each layer in the Transformer encoder includes Layer Normalization (LN) and residual (skip) connections

$$z' = \text{MSA}(\text{LN}(z)) + z \tag{6}$$

$$z'' = \text{MLP}(\text{LN}(z')) + z' \tag{7}$$

The Multi-Layer Perceptron (MLP) usually consists of two linear transformations with a Gaussian Error Linear Unit (GELU) [10] non-linearity in between. Assuming W_1 and W_2 are learnable weight matrices:

$$\text{MLP}(x) = W_2(\text{GELU}(W_1 x)) \tag{8}$$

Output Layer

The final output of the classification token is passed through a linear layer to produce the classification logits. Given C is the number of classes and $W_{class} \in \mathbb{R}^{C \times D}$:

$$\text{logits} = W_{class} \cdot z''_{class} \tag{9}$$

The linear layer projects the final representation of the classification token into the space of class labels.

2.2 Self-Supervised Feature Learning

Self-supervised feature learning is a crucial process that identifies patterns within extensive unlabeled datasets without the need for human-annotated labels. Plenty of research has been done in this field in the recent years. Several methods have been proposed, each with unique mechanisms and varying levels of success.

2.2.1 Contrastive Learning

Contrastive learning has gained significant attention for its effectiveness in self-supervised feature learning. One of the seminal works in this area is SimCLR [11]. It

employs a simple yet robust framework that leverages data augmentations to create positive pairs from the same image and negative pairs from different images. The model uses a contrastive loss to distinguish between these pairs, learning robust representations in the process. On the other hand, MoCo (Momentum Contrast) [12] introduces a dynamic dictionary with a momentum encoder. This approach allows the model to maintain a queue of negative samples, effectively reducing memory requirements and improving scalability. Nevertheless, it still requires a substantial number of negative samples to function optimally and necessitates careful tuning of the momentum parameter to balance stability and learning efficiency.

2.2.2 Clustering-Based Feature Learning

Clustering-based feature learning approaches automatically uncover the natural groupings of data within the latent representation space. This clustering process helps in understanding the inherent structure of the data by grouping similar data points together based on learned features. Agglomerative Clustering with Self-supervision [13] can capture multi-scale structures and is found to be effective for diverse datasets. However, it is found to be computationally expensive and needs careful tuning of the self-supervised task. SwAV [14] combines clustering with contrastive learning by swapping assignments between different augmented views of the image. This method is efficient in terms of computational resources and achieves state-of-the-art performance on several benchmarks. But it is sensitive to the choice of hyperparameters.

2.2.3 Distillation-Based Methods

Distillation-based methods have also shown considerable promise in self-supervised learning. Bootstrap Your Own Latent (BYOL) [15] introduces a teacher-student network where the student learns to predict the teacher's representations. Remarkably, BYOL achieves this without using negative samples, simplifying the training process and reducing computational demands. However, it is sensitive to the choice of data augmentations and network architecture, and there is a potential risk of model collapse if not properly tuned. DINO [4] extends the self-distillation approach to Vision Transformers [1]. DINO captures global image representations effectively without relying on negative samples. It shows strong performance on object detection and segmentation tasks, showcasing the potential of transformers in self-supervised learning.

Unlike traditional unsupervised representation learning methods that focus primarily on learning generalized visual features, our research centers on CutLER [3], which leverages these learned features, specifically DINO [4], to focus on the task of instance detection. While CutLER builds upon the robust feature representations provided by DINO, it further enhances performance through advanced techniques and refinements.

2.3 DINO

The self-supervised model DINO, introduced by Caron, Mathilde, et al. [4], achieves remarkable performance that rivals many state-of-the-art CNNs trained with supervision. DINO stands out for its ability to extract features that reveal clear information about semantic segmentation and scene layout within images. This capability distinguishes DINO from supervised ViTs and CNNs, underscoring its potential for sophisticated computer vision tasks without relying on annotated data.

As we will be using DINO features for producing the pseudo masks in CutLER [3], we need a basic understanding of DINO architecture and training.

2.3.1 Knowledge Distillation

Knowledge distillation plays a crucial role in training a student model to mimic the behavior and representations learned by a teacher model, both of which are ViTs.

Initially, the teacher model is a ViT that is pre-trained on a large dataset using self-supervised learning techniques. The teacher captures rich, generalized features from the data. The student model is a smaller ViT that aims to replicate the teacher's performance but with fewer parameters, making it computationally lighter and potentially faster during inference.

Momentum Encoder for Teacher

Instead of using the teacher model directly, DINO employs a momentum encoder mechanism for stability and improved generalization. This means that the parameters of the teacher model are updated using a moving average of the student model's parameters, rather than directly during training.

$$\theta_t \leftarrow m\theta_t + (1-m)\theta_s \tag{10}$$

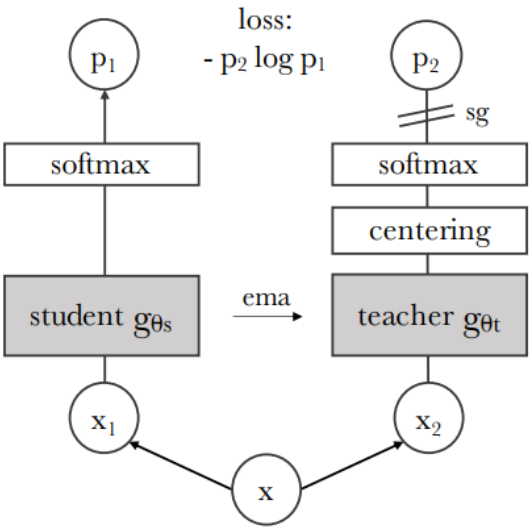

Figure 4: Architecture of DINO Illustration provided in [4].

The teacher model's parameters are updated using a momentum update rule as given in Eq. 10. Where θ_t are the parameters of the teacher model, θ_s are the parameters of the student model, and m is a momentum parameter (typically close to 1) that controls the rate of updating.

2.3.2 Training Process

DINO uses different augmentations of the same image to create multiple views. These augmented views are passed through both the teacher and student models. Outputs from both models are projected into a lower-dimensional space using projection heads. The optimization objective is to minimize the cross-entropy loss between the predicted probability distributions of the teacher and student models. Assume $P_t(x)$ and $P_s(x)$ represent the probability distributions predicted by the teacher and student models, respectively. The training process is illustrated in Fig. 4.

$$\min_{\theta_s} \mathcal{H}(P_t(x), P_s(x)) \tag{11}$$

The cross-entropy loss is computed between the softened distributions of the teacher and student models across all augmented views as given in Eq. 11.

2.4 Unsupervised Object Detection and Instance Segmentation

If we consider the recent methods for unsupervised object detection semantic segmentation, most of them leverage on self-supervised ViT [1] features. In DINO [4] it is observed that the underlying semantic segmentation of images can be extracted using the saliency maps from the ViT.

The quality of this segmentation surpasses existing methods when the image contains only a single instance. The effectiveness of DINO features in separating foreground and background has been affirmed by later works [16]. Building on this observation, both LOST [17] and TokenCut [5] utilize DINO features to segment a single salient object from each image. These methods capitalize on the strength of DINO to construct a graph from the features of image patches. Unlike TokenCut and DINO, which can only detect one instance, LOST is capable of finding multiple instances within an image. However, it can't be used as a pre-trained model for down stream tasks. CutLER [3] can not only detect multiple instances, but the model can also be further used as a pretrained model for label-efficient and fully-supervised learning.

MaskDistill [18] advocates a data-driven approach to mine object masks via self-supervised vision transformers and distill multiple object masks per image via an object segmentation model (Mask R-CNN [29]). Then a final segmentation model is trained using the found object masks. Although MaskDistill produces masks of similar quality to MaskCut, it generates only one class-agnostic mask per image. In contrast, MaskCut creates up to N masks per image to be used as pseudo labels. As a result, MaskCut has an advantage over MaskDistill in terms of quantity.

FreeSOLO [7] and the follow up work Exemplar-FreeSOLO [19] (with its addition of a randomly drawn pool of exemplars used in a contrastive learning loss) generate coarse segmentation masks with low guality and refine it further through self training similar to CutLER. But the poor quality of the coarse maps is a major drawback of this method, whereas CutLER masks made by the MaskCut [3, 5] algorithm are usually better in quality and quantity than the initial masks used by MaskDistill [18] and FreeSOLO [7].

Since CutLER outperforms other methods in most areas such as generating better pseudo ground truth masks, detecting multiple instances, and being compatible with various detection architectures, as well as serving as a pretrained model for supervised detection, our work primarily focuses on studying and enhancing the performance of

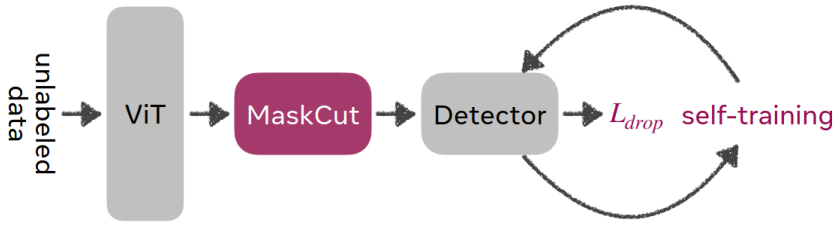

Figure 5: CutLER Overview. CutLER uses MaskCut for extracting coarse masks from the features of a self-supervised ViT. Following this, a detector utilizing a loss dropping strategy designed to be resilient against objects that MaskCut may overlook is used. Additionally, the model undergoes further enhancement through multiple rounds of self-training. Illustration taken from [3].

CutLER.

2.5 CutLER

CutLER [3] introduces a novel approach to address the challenges of object detection and instance segmentation in an unsupervised learning framework. By integrating Copy-Paste [20] augmentation and a contrastive learning framework, the method not only circumvents the need for labeled data but also achieves state-of-the-art results in object detection and instance segmentation. The complete process is illustrated in Fig. 5.

2.5.1 Normalized Cut

Normalized Cut (NCut) [21] is a graph cut method where the goal is to partition an image into segments (or regions) by cutting a graph that represents the image, such that the similarity within each segment is maximized while the dissimilarity between different segments is also maximized. Understanding the NCuts algorithm is crucial, as CutLER's pseudo-ground truth masks rely on repeated NCuts to generate accurate masks. This process is key to CutLER's improved performance in unsupervised instance detection and segmentation.

In Normalized Cut, an image is represented as an undirected graph $G = (V, E)$, where V is the set of nodes, each representing a pixel or a group of pixels and E is the set of edges, each representing a connection between two nodes, with weights

$w(i, j)$ indicating the similarity between nodes i and j. A basic Cut is defined as the sum of the edge weights that are severed by the partition as given in Eq. 12:

$$\text{Cut}(A, B) = \sum_{i \in A, j \in B} w(i, j) \tag{12}$$

However, minimizing the cut alone tends to produce small, isolated segments. To address this, the Normalized Cut criterion is introduced. The Normalized Cut is defined as follows:

$$\text{Ncut}(A, B) = \frac{\text{Cut}(A, B)}{\text{Assoc}(A, V)} + \frac{\text{Cut}(A, B)}{\text{Assoc}(B, V)} \tag{13}$$

where $\text{Assoc}(A, V) = \sum_{i \in A, j \in V} w(i, j)$ is the total connection from nodes in A to all nodes in the graph. The goal is to find the partition (A, B) that minimizes the NCut value. This can be expressed as an eigenvalue problem. The solution involves finding the eigenvector corresponding to the second smallest eigenvalue of the generalized eigenvalue problem given as Eq. 14.

$$(D - W)x = \lambda Dx \tag{14}$$

where $d(i) = \sum_j w(i, j)$ is the degree of node i, and D is a diagonal matrix with $d(i)$ on the diagonal and W is the weight matrix of the graph. The resulting eigenvector is used to partition the graph by thresholding its values.

2.5.2 TokenCut

TokenCut [5] builds on the principles of Normalized Cut but adapts them to work within the framework of ViTs. Instead of operating on pixels, TokenCut segments an image by working with tokens - image patches processed by a ViT. In TokenCut, each token is treated as a node in a graph, and the edges are defined by the self-attention scores from the transformer, which capture the affinity between tokens. The goal remains similar: to partition the graph of tokens in a way that minimizes the NCut criterion. By leveraging the transformer's attention mechanism, TokenCut can effectively capture global dependencies and segment objects in a self-supervised manner, even without labeled data. However, TokenCut only produces one mask per image, which limits the method from detecting multiple objects in an image. This issue is addressed in MaskCut.

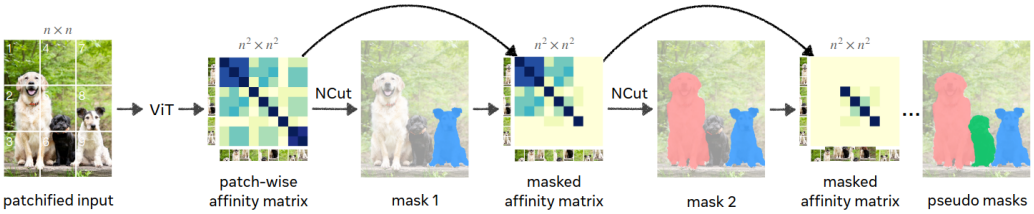

Figure 6: MaskCut works on the patch-wise similarity matrix for the image using a self-supervised DINO [4] model feature. N=3 defines the number of times NCut [21] is repeated on the background. In this case, 3 instances will be discovered per image in each step. Illustration taken from [3].

2.5.3 MaskCut

Like TokenCut [5], MaskCut considers the image segmentation problem as a graph partitioning task [21]. The nodes are tokens (each representing an image patch) and edges connect every pair of nodes, with weights W_{ij} reflecting the similarity between tokens. The optimization problem reduces the cost of dividing the graph into two sub-graphs, or a bipartition, by solving a generalized eigenvalue problem as given in Eq. 14.

The similarity weight W_{ij} of NCut in TokenCut is based on the similarity of patches in the DINO feature space. Following recent methods [22, 23, 24], they specifically employ the cosine similarity of 'key' features from the final attention layer of the DINO-pretrained model, represented as:

$$W_{ij} = \frac{K_i \cdot K_j}{\|K_i\|_2 \|K_j\|_2} \tag{15}$$

where K_i denotes the 'key' feature of patch i. They then solve Eq. 14 to find the second smallest eigenvector x. The main drawback of TokenCut is that it only uses the smallest eigenvector resulting in finding only one instance in the image. MaskCut overcomes this drawback and finds more instances by iteratively applying the same process in the background N times as given in Fig. 6. The figure shows the flow of MaskCut algorithm for N=3, where N defines the number of times NCut is repeated. In this case, up to 3 instances will be discovered per image.

Building on the work of [24, 25], a patch-wise similarity matrix for the image using features from a self-supervised DINO model [4] is created. Normalized Cuts [21] is applied to this matrix to obtain a single foreground object mask for the image. Subsequently, this foreground mask is used to mask out the affinity matrix values and repeat the process. This iterative approach enables MaskCut to identify multiple

object masks within a single image.

MaskCut uses two conditions to improve the performance.

1. An object centric prior [26] is used to filter out backgrounds. ie, if the foreground contains more than 2 out of 4 corners, foreground and background are switched.

2. From the intuition that foreground patches are more prominent than background ones [25, 27], we assert that the foreground mask should contain the patch corresponding to the maximum absolute value in the second smallest eigenvector.

If condition 1 is not satisfied and the current foreground contains two corners, background and foreground are switched.

2.5.4 DropLoss

A standard detection loss penalizes predicted regions r_i that do not overlap with the 'ground truth'. Since the ground truth masks from MaskCut may miss some instances, the standard loss does not allow the detector to identify new instances not labeled in the ground truth. To address this, the author proposes ignoring the loss for predicted regions r_i with minimal overlap with the ground truth.

$$L_{\text{drop}}(r_i) = \mathbb{1}(\text{IoU}_i^{\max} > \tau^{\text{IoU}}) L_{\text{vanilla}}(r_i) \tag{16}$$

Specifically, during training, the loss is dropped for any predicted region r_i that has a maximum overlap of τ^{IoU} with any ground truth instance as given in Eq. 16 where $\text{IoU}_i^{\max}$ denotes the maximum IoU with all ground truth for r_i, and L_{vanilla} refers to the standard loss function for detectors. L_{drop} avoids penalizing the model for detecting objects missed in the 'ground truth', thus encouraging the exploration of different image regions.

2.5.5 Copy-Paste Augmentation

Copy-Paste Augmentation [20] is a data augmentation technique used to enhance the diversity of training datasets by artificially creating new training samples. This method involves copying objects from one image and pasting them onto another, thereby generating new training examples with diverse object placements and backgrounds as illustrated in Fig. 7. The pasted objects can be resized, rotated, or otherwise manipulated to fit the new context. Lastly, the ground-truth annotations are also adjusted accordingly.

Figure 7: Illustration of Copy-Paste augmentation from [20].

In Cutler, instead of using the standard copy-paste augmentation, where masks are rescaled with a factor between 0.8 and 1.25, masks are randomly downsampled with a scalar uniformly sampled between 0.3 and 1.0. This approach enables Cutler to recognize even smaller instances in the image effectively.

2.5.6 Training

The training process is divided into two stages: initial training followed by self-training, as depicted in Fig.5. CutLER is agnostic regarding the choice of detector, allowing the use of any preferred detector. However, based on the experiments detailed in the paper, Cascade Mask R-CNN[28] yields better results compared to Mask R-CNN [29].

First, pseudo-ground truth masks are generated using MaskCut for all images in Imagenet [30] training set. The detector with a ResNet-50 [6] backbone is trained using these pseudo-ground truth masks for 160K iterations with Copy-Paste augmentations and DropLoss.

Self-Training

To further improve the model performance, several self-training loops are also carried out. As depicted in the Fig. 8, the pseudo-ground truth masks for the first round of self-training are formed by combining two sources. First, the CutLER mask predictions for each image with a confidence score greater than 0.7 are generated using the final model from the last training phase. Second, corresponding MaskCut masks that overlap less than 50% with the predicted masks are included. Together,

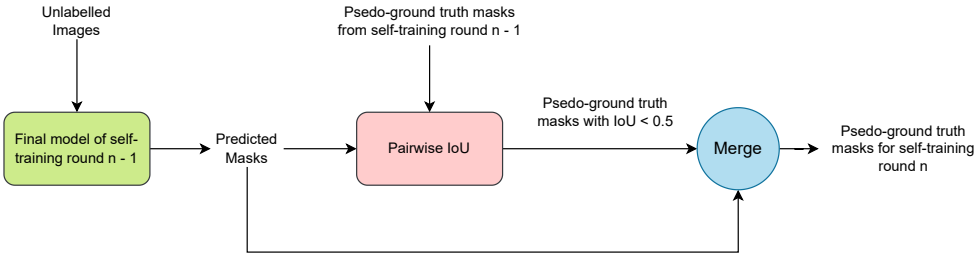

Figure 8: Mask Filtration in Baseline before each self-training loop

these masks form the pseudo-ground truth for the next phase. The mask refinement method is further explained in Section 2.5.7.

The same process is repeated for the following self-training rounds except that instead of MaskCut masks, pseudo-ground truth masks of the previous stage are used to compare with predicted masks. Each self training round consists of 80K rounds and does not use DropLoss as we obtain comparatively good quality masks in the first round itself. The detailed information about the training with Figures is explained later in Section 3.2.3 and 4.3.

2.5.7 Mask Refinement in CutLER

The presence of incorrect masks in pseudo-ground truth can lead to overfitting on incorrect patterns or failure to generalize properly across different instances. To mitigate these issues, techniques like iterative refinement, robust loss functions, and the incorporation of consistency constraints have been proposed. Tang et al. [31] and Wang et al. [32] explore these approaches.

Iterative refinement helps in progressively reducing this noise, leading to cleaner and more reliable labels [33]. Popular refinement methods incorporate strategies like thresholding, where only high-confidence predictions are used for retraining, or use ensemble methods to combine predictions from multiple models for more reliable masks. CutLER also uses this approach.

In CutLER, before each self-training loop, 30 masks per image are generated for the entire Imagenet dataset using the latest trained model. Of these, masks are filtered based on the confidence score. In the paper, the masks with confidence score greater than $0.7 - 0.05 * i$ on the i th iteration are kept and the rest are rejected. These filtered CutLER masks are compared with the corresponding MaskCut masks (in the first self-training round) or the pseudo-ground truth masks from the last round for each image. If the IoU between CutLER mask and MaskCut mask is less

than 0.5, the corresponding MaskCut mask is added along with CutLER masks and this constitutes the pseudo-ground truth for the self-training loop.

The intuition is to retain masks from previous pseudo-ground truths that do not significantly overlap (i.e., overlap less than 0.5) with the current predictions. This strategy allows CutLER to explore new regions of the image that have not been thoroughly examined in previous iterations. However, a challenge with this approach is that it may perpetuate the inclusion of noisy masks in the ground truth during each self-training loop. Our approach seeks to address this issue by implementing a more refined method for removing the noisy background masks from the MaskCut masks and to improve the quality of the masks iteratively, which will be explained in detail in Section 3.2.3.

3 Approach

In the field of unsupervised instance detection and segmentation, CutLER [3] gives a strong performance through exploiting an object-centric prior by training on ImageNet [30], as most images contain a single object in the center of the frame. Due to its strong instance discrimination abilities, CutLER is the current state-of-the-art method for this task.

In this chapter, we are exploring the limitation of CutLER, looking deeper into the special cases where CutLER fails such as overlapping instances and complex backgrounds. From the drawbacks of CutLER, we propose to train the model without overlapping instances, observing that overlapping instances are one of CutLER's main failure cases. We also investigate the hypothesis that a major reason for the better performance of CutLER is its object-centric prior. Furthermore, using the gathered information from the observations, we introduce a hypothesis to refine MaskCut masks using CutLER predictions to train the model from scratch to obtain a better evaluation score across a variety of datasets.

3.1 Assessing the Limitations of CutLER

Even though CutLER is the state-of-the-art model for unsupervised instance detection and segmentation, it still has several shortcomings. We go through some of them in this section.

3.1.1 Dependence on Initial Masks

CutLER relies on initial masks provided by MaskCut. If these initial masks are of poor quality, the performance of CutLER may be adversely affected. These pseudo-ground truths often contain inaccuracies due to imperfect initial segmentation, which can arise from factors like complex backgrounds, occlusions, and variations in object appearance. Such imperfections can mislead the model, causing it to learn incorrect features and boundaries, ultimately degrading the quality of instance detection and segmentation. As MaskCut produces the masks based on a hyperparameter N

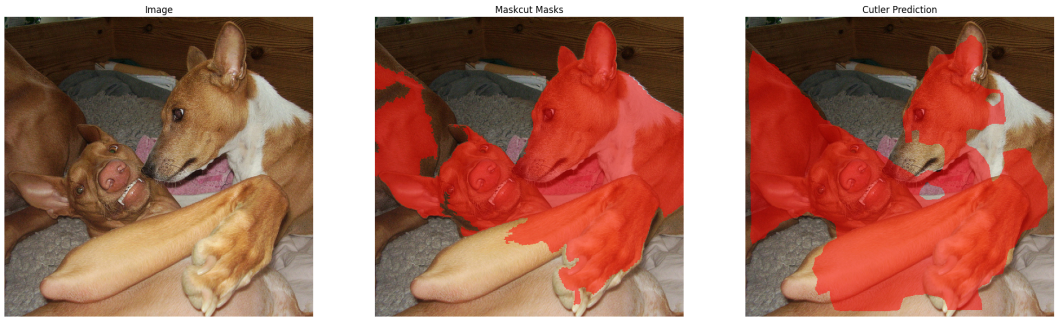

Figure 9: Dependence on Initial Masks reflected on the final CutLER prediction

(maximum number of masks generated per image), it also influences the quantity and quality of the pseudo-ground truths.

In challenging scenarios with cluttered backgrounds, occlusions or low-quality images, MaskCut could produce noisy masks as in Fig. 9 which can also affect the quality of CutLER predictions. One common method to filter out good masks is thresholding, which is used in the baseline and our approach during self-training.

To address this issue before self training, in our approach, we filter the MaskCut masks using the CutLER predictions and train from scratch, assuming the learning from scratch using better quality masks could improve the performance of the model. The approach is explained in detail in Section 3.2.3.

3.1.2 Overlapping Instances

Identifying instances using unsupervised instance detection or segmentation methods present significant challenges, especially when instances are closely positioned or overlapping in an image. Overlapping objects often blend together, making it difficult for the model to accurately segment and differentiate them as distinct entities. The lack of explicit supervision complicates the model's ability to learn and generalize the spatial relationships and boundaries between objects.

As illustrated in Fig. 10, in almost all cases of images with overlapping instances, MaskCut and CutLER group those instances together. MaskCut primarily separates the foreground from the background rather than distinguishing between individual instances [16]. This limitation is analyzed in Section 3.1.4. As a result, in Fig. 10, it can successfully detect the fish but not the group of people as there is an overlap between them. Consequently, in images with overlapping instances, both MaskCut and CutLER generate outputs that are more akin to semantic segmentation than true instance segmentation.

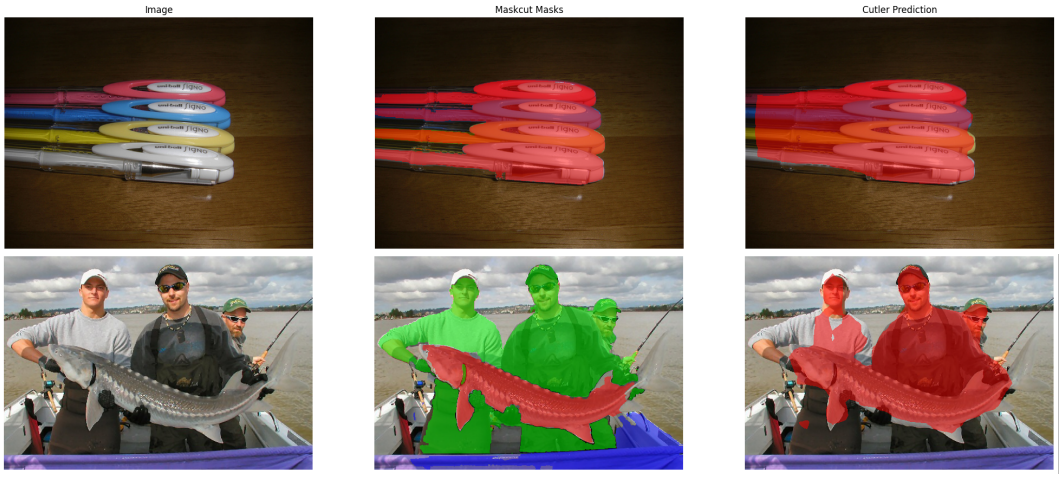

Figure 10: Grouping of Overlapping Instances in MaskCut and CutLER outputs, which is a common problem in most of the existing unsupervised instance detection methods.

The problem is not exclusive to MaskCut or CutLER, but most of the existing methods [16, 27, 34] are also affected by this issue. Solving this problem requires devising an algorithm that could use the instance-discriminating representations from DINO and use it to separate the instances. We attempted to address this issue by using Keypoint Correspondences, which is a challenging task. As we could not come up with a graph-cut algorithm that fits the problem, it is not a main part of this work. Nevertheless, it can be accessed in the Future Works Section 5.1.1.

Another solution to address the grouping of instances would be to provide explicit semantic information of instances during training. It is explored in Wang et al. (2023) [3] by testing with low-shot settings, ie, 2% and 5% labeled data, CutLER achieves 5.4% and 7.3% higher AP_{box} than the fully supervised MoCo-v2 with better separation of close and overlapping instances. But as our approach focuses solely on improving the unsupervised instance detection performance, the problem of grouping instances remains unsolved in our approach as well. But we intent to explore the influence of overlapping instances on CutLER training and evaluation which is explained in detail in the Section 3.2.1.

3.1.3 Images with Complex Background

Even though DINO features are good at providing foreground-background separation, images with complex or unusual backgrounds can possess challenges to this task. In our approach, we intent to address the issue of unwanted masks generated due to

Figure 11: Images With Complex Backgrounds impacting MaskCut and CutLER outputs, leading to undesired background mask generation

complex backgrounds as shown in Fig. 11. Such masks can not only be present in the MaskCut masks which act as the pseudo-ground truth for CutLER training, but also in the CutLER predictions itself. As per our observation, the mask filtration strategy used by the baseline doesn't address this issue.

In CutLER, a self-training loop is implemented to iteratively refine the pseudo ground truth masks. We hypothesize that removing undesired background mask before training and self-training phases could improve the performance of the model. We plan to test this hypothesis by modifying the mask filtration algorithm used to refine pseudo-ground truth masks in the baseline method before each self-training loop. This modified algorithm will be used to generate improved MaskCut masks, enabling us to train the model from scratch with better pseudo-ground truth. Our approach addressing this issue is explained in Section 3.2.3.

Limitation of CutLER Mask Filtration - Qualitative Examples Figure 12 illustrates some qualitative examples of how the baseline filters masks for the next round of training. Despite having either good MaskCut mask or CutLER prediction, baseline selects large incorrect MaskCut masks which don't overlap much along with CutLER prediction masks, which adversely affect the model performance. We intent to remove these false masks to generate better quality pseudo-ground truths.

3.1.4 Suboptimal Use of DINO Features in MaskCut

MaskCut is the pseudo-ground truth mask generation method for CutLER. Even though MaskCut proves to be better than other existing methods like TokenCut [3],

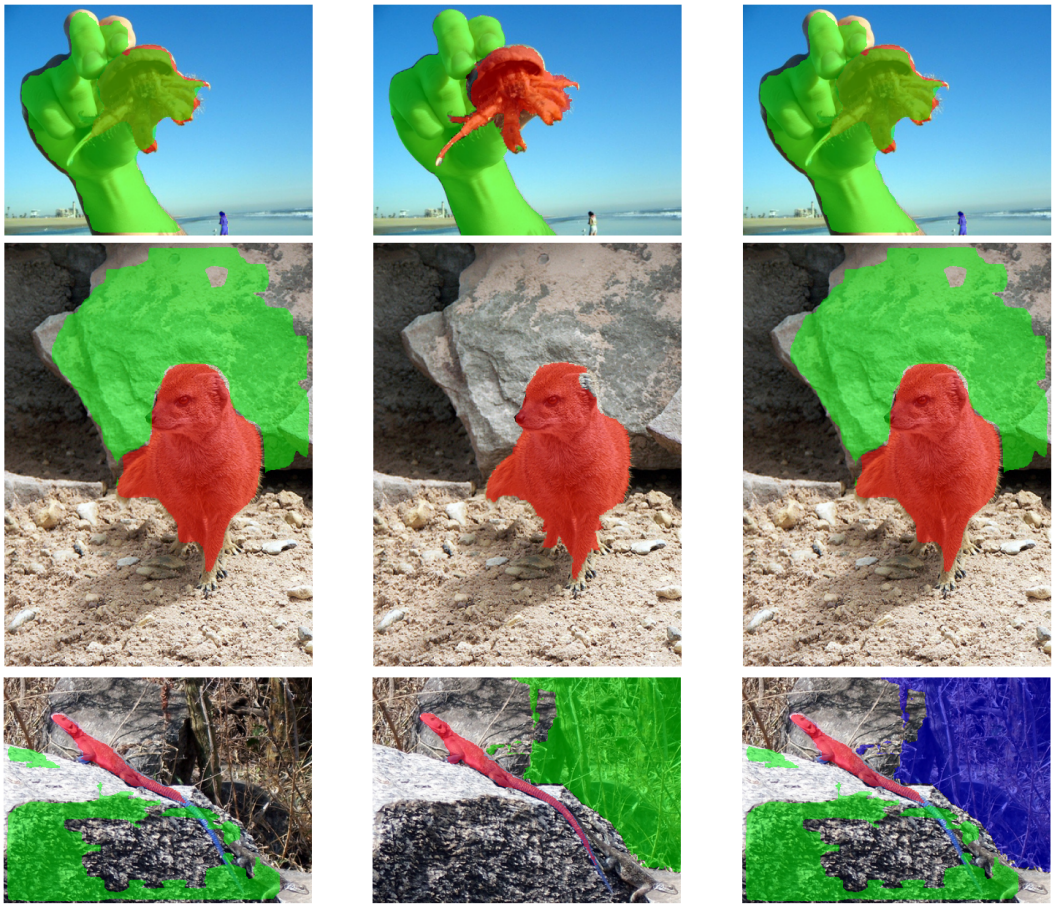

Figure 12: Mask Filtration in Baseline - Qualitative Examples Examples illustrate CutLER prediction masks, MaskCut masks and masks selected by CutLER mask filtration method (left to right)

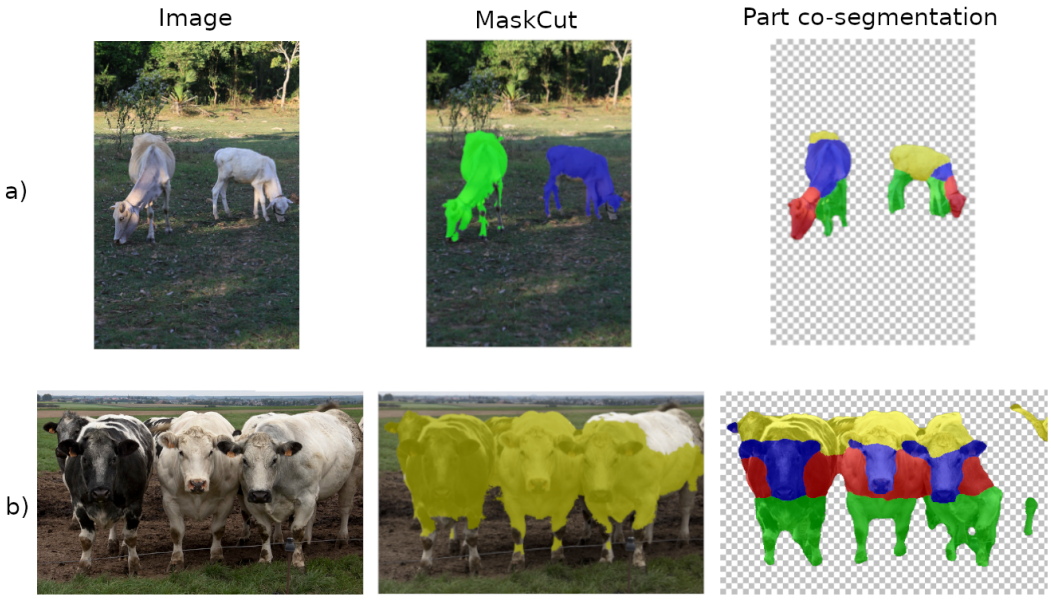

Figure 13: MaskCut vs Part Co-Segmentation Masks. Discriminative features that could help differentiate instances are present in DINO features, as evidenced in n-part co-segmentation. But these features are not leveraged by MaskCut.

it still comes with limitations.

MaskCut is not effectively making use of the discriminative DINO features that could help differentiate the instances. As it can be observed from Fig. 13, the part co-segmentation map generated by clustering feature maps obtained from the self-attention layers of DINO clearly contains features that could help discriminate these instances. However, MaskCut does not fully utilize these features. Instead, it operates more on a coarse level by segmenting the image into foreground and background regions.

MaskCut also comes with a fixed hyperparameter N, which determines the maximum number of masks to be produced per image, regardless of the actual number of distinct instances present. This constraint limits its flexibility and effectiveness, especially in complex scenes with multiple instances. As a result, MaskCut often under-segments instances even though the instance discriminative features are provided by DINO.

3.2 Proposed Approaches

3.2.1 Approach to Estimate the Impact of Overlapping Instances

Based on the observation that MaskCut often groups overlapping instances into a single mask from Section 3.1.2, we hypothesize that training CutLER on ImageNet images without overlapping instances may improve performance compared to training on all images. This approach could reduce the number of inaccurate masks (removing grouped instance masks), improving the overall quality of pseudo-ground truth masks. Additionally, we aim to explore whether a model trained exclusively on non-overlapping instances can enhance detection of individual instances in images where overlaps occur.

For the sake of completeness and to observe whether there is any relative improvement or loss, we compare three approaches of training CutLER.

1. Using all images of ImageNet (same as the baseline)
2. Using images without any overlapping instances
3. Only using images with overlapping instances

We expect that the model trained without overlapping instances will outperform the baseline, while the model trained exclusively on images with overlapping instances will likely underperform compared to the baseline. Splitting the evaluation dataset

according to these three approaches should also reveal a similar performance pattern. Our hypotheses are supported by the positive results from the experiments in Section 4.4.1.

To test the hypothesis, we make use of ground truth bounding box annotations provided by ImageNet and the images with an overlap (IoU) of $\tau > 0.1$ are taken as the criteria to filter images for approach 2 and 3. As ImageNet contains mostly object-centric single instance images, the third approach (only using images with overlapping instances) would have a significantly lower number of training images compared to the other two approaches. Specifically, Approach 3 utilizes 6% of the annotated ImageNet dataset, while Approach 2 uses the remaining 94%. Given the significantly smaller training sample size in Approach 3, a fair comparison isn't possible. Therefore, we focus on Approach 2 for comparison with the baseline.

Through this approach, we expect to observe an improvement by using less training data. But using this method in unsupervised fashion is rather difficult. Due to the grouping of nearby instances, the process of filtering images with overlapping instance is extremely challenging. Hence these tests are carried out using bounding box labels of ImageNet. However, the approach gives insights on the influence of overlapping instances in training that can be useful for future research.

3.2.2 Approach to Estimate the Impact of Object-Centric Instances

To support our hypothesis that an object-centric prior is a key contributor to CutLER's state-of-the-art performance, we propose that filtering out non-object-centric images from the training dataset could further enhance its effectiveness. Recent research has highlighted the significant role that object-centric images play in model performance [16, 35].

Object-centricity is hard to define. As we couldn't find a standard way to define object-centicity from our research [36, 35], we defined two simple hyperparameters to filter object-centric images.

- **Area Threshold** : Minimum area ratio of bounding box to image
- **Center Threshold** : Maximum distance ratio from the image center to bounding box center to the image diagonal.

We set Area Threshold to 0.1 and Center Threshold to 0.2. That is, if bounding box has size less than 10% of the size of the image or the distance between image center and bounding box center is more than 20% of image's diagonal length, that image will be rejected from the training set.

Like in the previous section, to test the hypothesis, we make use of ground truth bounding box annotations provided by ImageNet. We are only using single instance images as object-centric images can contain overlapping instances as well. Due to the filtration process, we only use 77% images of annotated ImageNet dataset, that is only 30% of total ImageNet images. Even with this much less data, we expect to observe a competitive result compared to the baseline.

3.2.3 Approach for Mask Filtration

Considering the problems with complex backgrounds and limitations of CutLER Mask Filtration method in Section 3.1.3, we propose a new mask filtration method to address these issues. In this section, we also provide a detailed explanation of the baseline (CutLER) mask filtration method, making it easier to understand the improvements offered by our approach.

CutLER Mask Filtration Method

To improve clarity and facilitate comparison with the proposed method, we break down the CutLER training process into two distinct phases: initial training and self-training, as illustrated in the Fig. 14.

Initial Training Initially, the detector (Cascade Mask RCNN) trained with ImageNet dataset using MaskCut masks as pseudo-ground truth for 160K iterations with Copy-Paste augmentations and DropLoss. As illustrated in Fig. 14, after the initial training, the trained model predicts masks for each image in ImageNet dataset (30 masks per image) using the trained model. Out of the 30 predicted masks, high-quality ones are filtered by applying a confidence score threshold of 0.7 and passed onto the mask filtration pipeline.

Mask Filtration and Self-Training Baseline mask filtration process is illustrated in Fig. 15. Pairwise IoU is calculated between the selected predicted masks and MaskCut masks. For mask pairs with an IoU below 0.5, the corresponding MaskCut masks are included, along with all selected predicted masks, to form the pseudo-ground truths for the next self-training stage. Intuitively, MaskCut masks that have less than 50% overlap with the selected predicted masks are included alongside the CutLER masks to form pseudo-ground truths for the next self-training round.

The goal is to retain as many non-overlapping masks as possible. However, always including masks that don't overlap with CutLER prediction masks can introduce

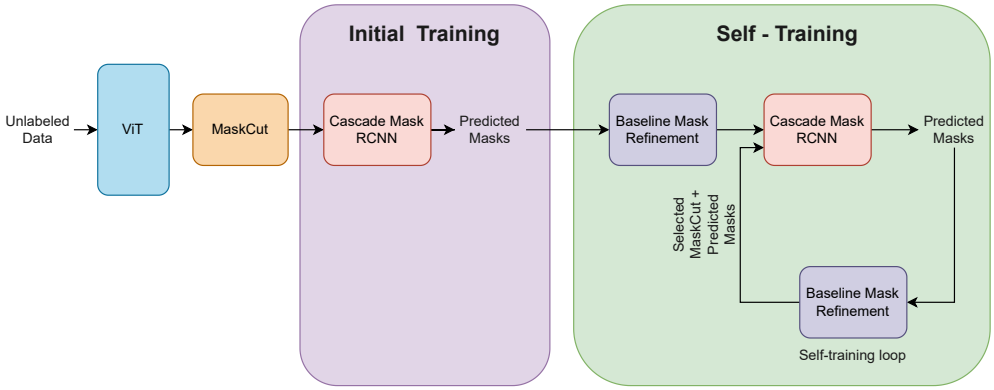

Figure 14: Baseline Training Pipeline with repeated mask filtration and self-training.

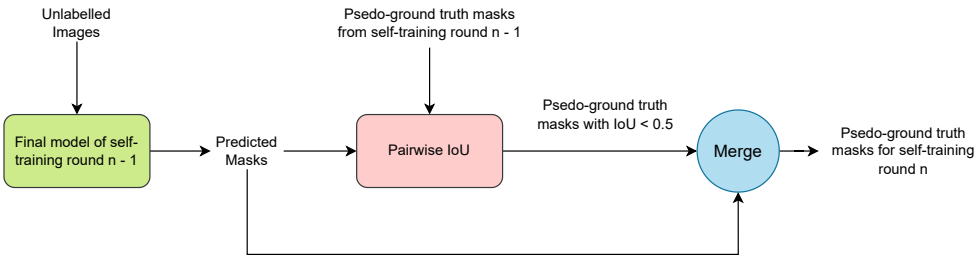

Figure 15: Mask Filtration in Baseline before each self-training loop.

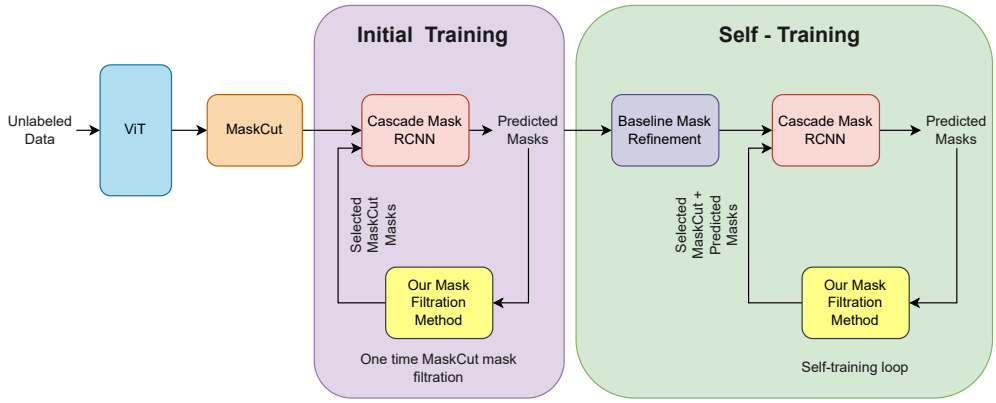

Figure 16: Proposed Training Pipeline 1 featuring a one-time proposed MaskCut mask filtration followed by multiple self-training loops with our mask filtration method.

irrelevant or unwanted masks into the pseudo-ground truth. This can affect the performance of the model.

For further self-training loops, the same procedure repeats, except that instead of MaskCut masks, pseudo-ground truth masks of the last round are used to compare with the predicted CutLER masks. Performance of the model claims to have improved up to 3 self-training loops by the authors. We will be running 2 self-training loops for both the baseline and proposed method in our experiments.

Proposed Mask Filtering Method

Emphasizing quality over quantity, we introduce an improved approach for mask filtration. Noting that the current mask filtration method in CutLER tends to include unwanted background masks in its pseudo ground truths, we propose to enhance the process by removing ambiguous masks from the ground truth instead of retaining them. This adjustment aims to improve the overall quality and reliability of the pseudo ground truths, leading to better model performance.

Initial Training As illustrated in Fig. 16, we train the detector for 160K iterations using Copy-Paste augmentations and DropLoss identical to the baseline. After which we introduce an additional step to refine MaskCut masks. This refinement involves preserving only high-certainty masks by comparing them against CutLER predictions.

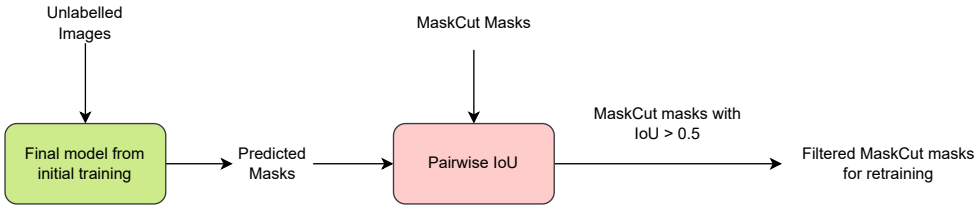

Figure 17: Mask Filtration in Our Method, used to filter MaskCut masks after the initial training.

Proposed Mask Filtration and Retraining Proposed mask filtration method is illustrated in Fig. 17. After the first training phase, like in the baseline, high-quality predicted masks are filtered by applying a confidence score threshold of 0.7. Instead of creating the new pseudo-ground truth by selecting masks from both MaskCut masks and CutLER prediction masks, we focus solely on filtering MaskCut masks. Rather than selecting MaskCut masks corresponding to mask pairs with an IoU $<$ 0.5 from the batch IoU matrix, we choose MaskCut masks that correspond to mask pairs with an IoU $>$ 0.5.

The idea is to filter highly certain MaskCut masks and discard possible incorrect masks. These selected MaskCut masks are treated as the new pseudo-ground truth and we train from scratch for 160K iterations with Copy-Paste augmentations and DropLoss. We are effectively repeating the same initial training process, but with better masks. With the filtered high quality masks in hand, we expect to achieve a better performance.

It's important to note that if no masks are selected for an image, that image is removed from the training set, resulting in a smaller dataset and reducing the training time (around 130K images are dropped from ImageNet during this stage). This approach effectively eliminates potentially unwanted masks from the pseudo-ground truth, possibly providing more accurate mask predictions.

Further Mask Filtration and Self-Training For self-training, we follow training pipeline of the baseline, training for 80K iterations without using DropLoss. We use our mask filtration method illustrated in Fig 18 to filter masks between self-training rounds.

We evaluate our approach in two scenarios: applying our mask filtration method versus the baseline mask filtration method during the self-training round. It is because the baseline mask filtration method may offer greater potential for further improvement across self-training loops, as it contains more masks, encouraging

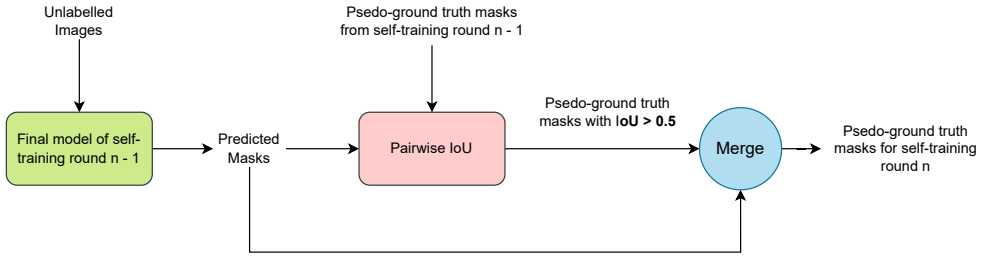

Figure 18: Mask Filtration in Proposed Method, used to filter masks during self-training.

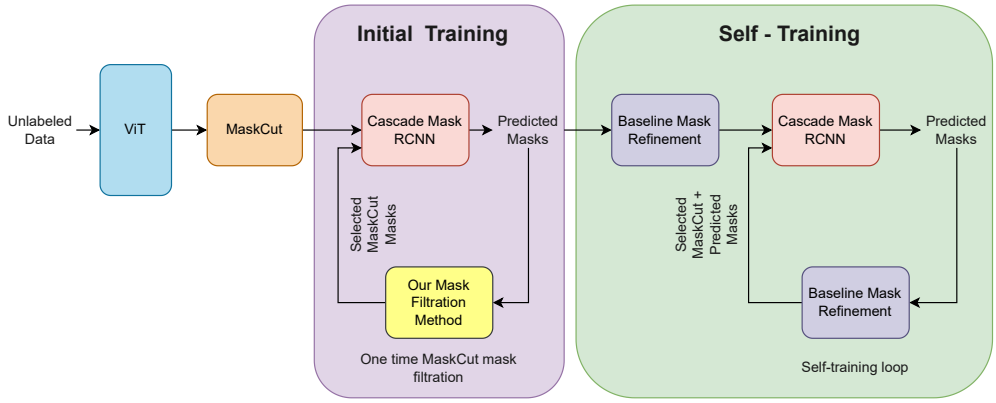

Figure 19: Proposed Training Pipeline 2 featuring a one-time proposed MaskCut mask filtration followed by multiple self-training loops with CutLER mask filtration method.

broader exploration.

Using Our Mask Filtration Strategy for Self-Training The only modification to the baseline approach in self-training loop is selecting pseudo-ground truth masks with IoU > 0.5 with predicted masks, rather than those with IoU < 0.5. This way, we always retain masks with high certainty.

But limiting pseudo-ground truth labels to only high-certainty masks could constrain improvements in the self-training round. This hypothesis is verified by the experiments carried out using our mask filtration method in self-training and can be accessed in Section 4.4.3.

Using CutLER Mask Filtration Strategy for Self-Training Considering the possible issues of employing our mask filtration method in the self-training loop as mentioned

in the last section such as limiting exploration and having few masks compared to the baseline, we also intent to test the combination of both.

As our proposed mask filtration method is rather conservative, we propose to use masks generated using our mask filtration method for initial training and baseline mask filtration strategy for self-training. This way, we can provide more refined masks for initial training and still encourage exploration during self-training.

The training pipeline is illustrated in Fig. 19. The only change is replacing our mask filtration method with the CutLER mask filtration method in the self-training part. This method showed the best performance among the proposed unsupervised approaches, with detailed experimental results provided in Section 4.4.3.

Recall Reduction Concerns By utilizing a smaller number of more accurate masks, we anticipate an improvement in precision. However, there may be a slight decrease in recall if the reduced number of masks does not sufficiently cover all ground truth instances. But our experiments indicate that this change is negligibly small. Detailed results and analysis on this can be found in the Section 4.4.5.

4 Experiments and Results

In this chapter, we present a comprehensive analysis of the experiments conducted to compare our proposed method with the baseline approach, CutLER. We thoroughly evaluate both models across a diverse set of datasets to assess their performance. Additionally, we delve into the impact of training images containing overlapping instances, providing detailed quantitative results to illustrate how these images affect the model's performance.

4.1 Datasets

For a fair comparison, we use the same datasets as the baseline for both training and evaluation. All models are trained on the ImageNet dataset and evaluated on a diverse set of benchmark datasets, including COCO, Pascal VOC, and KITTI. This ensures more consistent and comprehensive assessment of performance across different types of datasets.

4.1.1 ImageNet

The ImageNet dataset [30] is a large-scale visual database designed for use in visual object recognition research. Developed by researchers at Princeton and Stanford, it contains more than 10,000,000 labeled images depicting 10,000+ object categories. Each image in the dataset is hand-labeled by humans, making it a valuable resource for training and benchmarking deep learning models in computer vision.

We generate MaskCut annotations for all images on the subset of ImageNet containing the 1000 categories and 1.3 million images (ImageNet-1K), which serve as the pseudo-ground truth for our experiments. Both the baseline method (CutLER) and our proposed method are trained on the ImageNet dataset. However, in the proposed method, a fraction of images are excluded during the mask-filtration process (images with no annotaions are removed).

4.1.2 COCO

The COCO (Common Objects in Context) dataset [37] is a widely-used benchmark in the field of computer vision, designed to spur advancements in object detection, segmentation, and captioning. It contains over 200,000 images with more than 80 object categories, annotated with precise bounding boxes, segmentation masks, and context-related captions.

We use the validation set of the COCO 2017 split, which contains 5,000 images, for evaluating the models. Both bounding box coordinates and segmentation annotations are utilized as ground truths for evaluation.

4.1.3 PASCAL VOC

The PASCAL VOC 2012 dataset is a widely recognized benchmark in visual object recognition, comprising 11,530 images across 20 categories with comprehensive annotations for object detection, classification, and segmentation tasks. For evaluation, we use both the training and test images from the PASCAL VOC dataset and detailed bounding box annotations as ground truths.

4.1.4 KITTI

The KITTI dataset [38] is a prominent benchmark for evaluating performance in autonomous driving and computer vision tasks, including object detection, tracking, and scene flow. It features high-resolution images captured from a stereo camera setup mounted on a moving vehicle, encompassing a variety of urban and rural driving scenarios.

Although the KITTI dataset offers rich annotations, including 3D object labels and depth information, our evaluation focuses solely on bounding boxes. Since the dataset does not provide segmentation annotations, we utilize only the bounding box data to evaluate 7521 images from KITTI's trainval split.

4.1.5 Comic and Watercolor

In addition to real-world image datasets, we also incorporate art datasets, such as Comic and Watercolor [39], to evaluate the model's generalization capabilities across diverse visual styles. Since these datasets lack segmentation annotations, we use only the bounding box data for evaluation.

4.2 Metrics

We mostly use precision metrics to evaluate the performance of the models in our work. This includes metrics like AP and AP50, which are standard metrics used in object detection and instance segmentation tasks to evaluate the performance of models. These metrics provide a measure of how good a model is at correctly identifying and localizing objects within an image.

4.2.1 Average Precision

Average Precision (AP) is a metric that summarizes the precision-recall curve, which plots precision against recall at different confidence thresholds. Precision is defined as the ratio of True Positive (TP) detections to the sum of True Positive and False Positive (FP) detections, while recall is the ratio of True Positive detections to the sum of True Positive and False Negative (FN) detections. Precision and Recall is defined as Eq. 17 and Eq. 18 respectively.

$$P = \frac{TP}{TP + FP} \tag{17}$$

$$R = \frac{TP}{TP + FN} \tag{18}$$

The AP is calculated as the area under the precision-recall curve, which is typically computed using a numerical approximation method like the trapezoidal rule. It can be formally defined as Eq. 19, where n refers to different recall levels and R_n andP_n are the recall and precision at the n^{th} threshold.

$$\text{AP} = \sum_{n=1}^{N} (R_n - R_{n-1}) \cdot P_n \tag{19}$$

AP is typically averaged over multiple IoU thresholds. In our case, AP is averaged over IoU thresholds from 0.5 to 0.95 with a step size of 0.05.

AP50

AP50 is a specific case of the Average Precision metric, where the Intersection over Union (IoU) threshold is set to 0.50. IoU is a measure of the overlap between the predicted bounding box and the ground truth bounding box, defined as Eq. 20.

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}} \tag{20}$$

AP50 calculates the AP but only considers a detection as a true positive if the IoU between the predicted bounding box and the ground truth is greater than or equal to 0.50. This metric is useful for understanding how well a model can detect objects with a certain level of spatial accuracy.

4.2.2 Average Recall

Average Recall (AR) is a metric that measures the recall of a model averaged over multiple IoU thresholds. It quantifies how much of ground truth labels were accurately predicted by the model. It is a valuable metric for unsupervised detection as it does not penalize the models for detecting novel objects which are unlabeled in the dataset. The formula for AR is given by Eq. 21.

$$AR = \frac{1}{|T|} \sum_{t \in T} R(t) \tag{21}$$

where T is the set of IoU thresholds and $R(t)$ is the recall (Eq. 18) at a specific IoU threshold t. In our case, AR is averaged over IoU thresholds from 0.5 to 0.95 with a step size of 0.05.

AR100

AR100 (AR@100) is a variant of Average Recall that limits the number of detections per image to a maximum of 100. It represents the recall averaged across a range of IoU thresholds, constrained to 100 predictions per image. The formula for AR@100 is given by Eq. 22.

$$AR100 = \frac{1}{|T|} \sum_{t \in T} R_{100}(t) \tag{22}$$

where $R_{100}(t)$ is the recall at IoU threshold t, considering up to 100 detections per image. We use AR100 metric to compare recalls of the baseline and our model in Section 4.4.5.

4.3 Implementation Details

Our implementation largely follows the baseline approach; however, it is important to note a key difference in our setup. While in the baseline paper experiments use a batch size of 16, we utilize batch sizes of 4 and 8 due to resource constraints. To ensure a fair comparison, we also train the baseline model from scratch using these

same batch sizes of 4 and 8. All our experiments follow the following setting. If there are some modifications, it would be mentioned in the respective sections.

4.3.1 Training Data

Only the images from ImageNet dataset (1.3 Million images) are used for the initial training and self-training. We do not use any supervised pre-trained models or labels for training baseline or the proposed method. However, the bounding box annotations are used to analyze the impact of images with overlapping instances and object-centric prior in Sections 4.4.1 and 4.4.2 respectively.

4.3.2 MaskCut

We apply MaskCut with N=3, generating up to three masks per image through repeated N-Cut operations, on images resized to 480×480 pixels to create pseudo-ground truths. The value of N is optimal at 3 for generating best quality masks for ImageNet dataset [3]. The patch-wise affinity matrix generated from the key descriptors of the ViT-B/8 DINO model is used to perform the N-Cut operation. Additionally, we employ Conditional Random Fields (CRF) [40] to refine the masks and extract their bounding boxes.

4.3.3 Detector

Although CutLER is designed to be agnostic to the choice of object detector, we chose to use Cascade Mask R-CNN for all our experiments. This decision is based on the baseline paper's findings, which demonstrated that Cascade Mask R-CNN outperforms Mask R-CNN. We train the detector on ImageNet with MaskCut pseudo masks and bounding boxes for 160K iterations with a batch size of 4 or 8.

The Copy-Paste augmentation is also used during the training process to improve robustness of object detection and segmentation models by exposing them to a wider range of scenarios and object contexts. In order to detect small objects, instead of vanilla copy-paste augmentation, masks are randomly downsampled with a scalar uniformly sampled between 0.3 and 1.0.

We optimize the Detector using Stochastic Gradient Descent (SGD) for 160K iterations with a learning rate of 0.005, weight decay of 5×10^{-5} and a momentum of 0.9. Training follows a learning rate schedule which decreases it by 5 after 80K iterations.

4.3.4 Self Training

In the baseline, at each stage, along with CutLER mask predictions with confidence score > 0.7 generated using the model from the previous stage, Maskcut masks which have IoU < 0.5 with the CutLER prediction masks together make the pseudo-ground truth masks for that stage. This has already been explained in detail in Section 3.2.3. The detector is then optimized using SGD with a learning rate of 0.01 over 80,000 iterations. We do not employ DropLoss during these self-training phases just as in the baseline.

4.3.5 Resources

Generating MaskCut annotations for all images in ImageNet is supposed to be the most time consuming part. But we used the pre-generated MaskCut annotations to save time.

Initial training on ImageNet with batch size 8 spans over 160K iterations on four NVIDIA rtx-2080 gpus takes around 1 day 18 hours and self-training of 80K iteration takes around 21 hours. The Training using filtered MaskCut masks generated by our method takes 4 hours less (1 day 14 hrs) as around 130K images are dropped in the mask filtration step for not having any pseudo-ground truth masks.

Filtered non-overlapping instance images and object-centric images representing only 35% and 30% of the ImageNet dataset, respectively, could be processed more efficiently compared to the baseline. Both methods require only approximately 18 hours for training.

4.4 Experiments

4.4.1 Exploring the Impact of Overlapping Instances

In this section, we describe the experiments conducted to evaluate the performance of the approaches mentioned in Section 3.2.1. ie, 1) Using all images of ImageNet (same as the baseline). 2) Using images without any overlapping instances, 3) Only using images with overlapping instances. The primary goal is to assess how the presence or absence of overlapping instances in the training data influences the performance of the CutLER model. But we exclude approach 3 as we have insufficient images satisfying the condition (6% of the annotated dataset), hence a fair comparison is not possible.

Dataset

We use the ImageNet dataset for training, focusing on the subset with ground truth bounding box annotations. For training the baseline, the entire ImageNet dataset is used and doesn't use any bbox annotations. Our models trained with proposed mask filtration methods also use the data, except that some images (10%) are filtered out during mask filtration after the initial training.

For our approaches to analyze the impact of overlapping instances and object-centric prior, we utilize the annotated subset of ImageNet, which comprises 38% of the entire dataset. Within this subset, we filter out images for respective experiments. Consequently, our approach uses only 35% of the images employed by the baseline for the overlapping instance experiment and 30% for analyzing the impact of the object-centric prior. It's important to note that while bounding box annotations are used solely for filtering images, the training process itself remains unsupervised, just like the baseline.

We evaluate our approaches and the baseline using the COCO 2017 Validation dataset, which contains 5,000 images spanning 80 different classes. Precision is calculated based on both bounding box and segmentation ground truths. To gain deeper insights, we further split the evaluation dataset into two subsets: images with overlapping instances and those without. This allows us to better analyze whether training without images that contain overlapping instances can improve the model's ability to distinguish individual instances in images where overlap occurs. Given the diverse range of images in the COCO 2017 Evaluation dataset, the split between images with and without overlapping instances is more balanced compared to ImageNet, with 48% of the images containing overlapping instances and 52% without. This balanced split ensures a fair and comprehensive comparison between our approach and the baseline.

Training Procedure

The training procedure for both the baseline and our proposed methods follow the standard CutLER training pipeline. MaskCut mask annotations serve as the pseudo-ground truth, and a Cascade Mask R-CNN is employed as the detector. Training is conducted incorporating Copy-Paste augmentations and DropLoss with batchsize 4 and 8 to minimize resource requirements. The baseline is trained over 160K iterations, while our methods to analyze the impact of overlapping instances and object-centric prior require only 80K iterations due to the smaller dataset size. This adjustment reflects the reduced training data in our approach, allowing for a

more efficient training process without compromising the effectiveness of the model.

Metric	Baseline		Ours	
	AP	AP50	AP	AP50
bbox	10.74	19.78	**11.52**	**20.67**
segm	8.2	16.62	**8.80**	**17.47**

Table 1: AP and AP50 for Object Detection and Instance Segmentation Tasks on COCO validation set using models trained (without self-training) on all images and images without overlapping instances

COCO Val	Metric	baseline		Ours	
		AP	AP50	AP	AP50
All images	bbox	10.74	19.78	**11.52**	**20.67**
	segm	8.2	16.62	**8.80**	**17.47**
No overlapping instances	bbox	23.45	38.11	**24.77**	**39.46**
	segm	19.19	35.19	**20.13**	**36.36**
Only overlapping instances	bbox	7.25	15.11	**8.52**	**16.8**
	segm	5.12	11.55	**6.18**	**13.27**

Table 2: AP and AP50 for Object Detection and Instance Segmentation Tasks Evaluated on Split COCO Validation Set using models trained (without self-training) on all images and images without overlapping instances

Results

Even though the dropped images with overlapping instances are only 5% of the annotated ImageNet subset, there is a notable improvement in the results. As it can be observed from Table 1, our method improves in both object detection and instance segmentation tasks. The table shows AP and AP50 of both object detection (bbox) and instance segmentation (segm) tasks on COCO validation set evaluated using the baseline and our approach. Even though the experiment is conducted using batch size 4, the results are still comparable to the baseline and model trained using our proposed mask filtration method (will be explained later in Section 4.4.3) with batch size 8, which can be observed from Table 6. It is to be noted that the evaluation is performed using class-agnostic annotation generated from the original

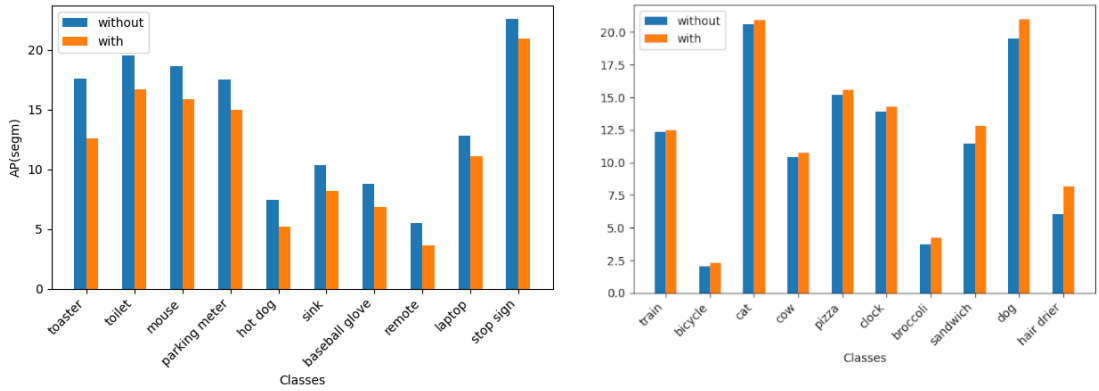

Figure 20: Class-wise Comparison of change in AP_{segm} of models trained using images with and without overlapping instances evaluated on COCO validation set - 10 most and least improved classes

COCO annotations. We reuse the same annotation file used in CutLER.

To gain deeper insights, we further split the evaluation dataset into two subsets: images with overlapping instances and those without. It is to observe to what extent our approach helps to detect individual instances from images with overlapping instances. Results can be observed from Table 2. As anticipated, the evaluation scores for images with overlapping instances are significantly lower compared to those for images without overlaps. This disparity is likely due to the inherent challenge of grouping of instances when they overlap. Despite this challenge, it is noteworthy that both splits - images with and without overlapping instances showed a nearly equal improvement in performance for instance detection and segmentation tasks. This suggests that our approach is effective across different scenarios, affirming the model's ability to generalize even in complex cases where instances are closely packed or overlapping.

Improvement Across Classes

To ensure that the observed improvements in performance were not simply due to the loss of accuracy in one class being offset by gains in another, class-wise improvements are calculated on the COCO validation set. This analysis allows for a more granular understanding of the model's performance across different classes. By evaluating each class individually, we can verify that the enhancements in instance detection and segmentation are consistent and not merely the result of compensatory effects between classes. This class-wise assessment provides a clearer picture of the model's

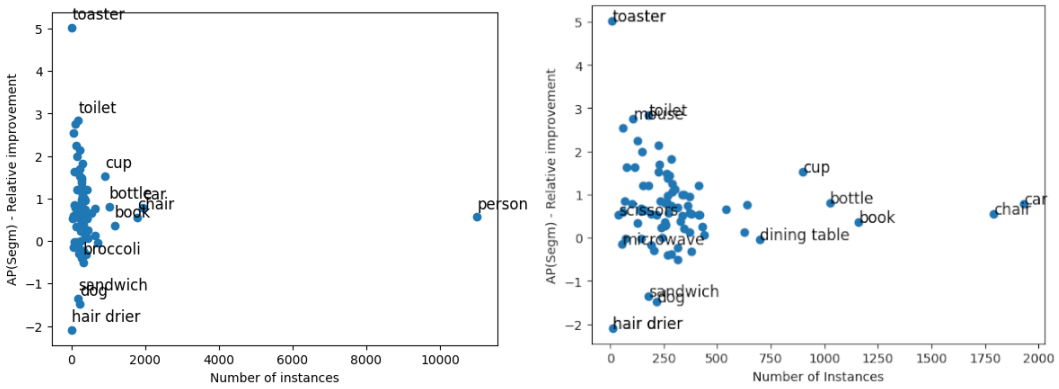

Figure 21: Relative Improvement (AP (Segm)) Across Classes Against Number of Instances. Relative improvement of our model against the baseline on the y-axis and number of instances on the x-axis. Classes corresponding to the extreme data points are highlighted. The plot on the right excludes the class **person** to make the analysis easier and more focused on the remaining classes.

true capabilities, ensuring that the overall performance gains are robust and evenly distributed across the dataset.

The test results are illustrated in Fig. 20, which highlights the 10 classes with the most significant improvements and the 10 classes with the least improvements evaluated on COCO validation set, based on the Average Precision for instance segmentation. Notably, our approach outperforms the baseline across the majority of classes. Specifically, it performs better in 68 out of the 80 classes assessed. In contrast, for the remaining 12 classes, our method shows reduced performance relative to the baseline.

To provide more clarity, we plot the relative improvement of our model over the baseline against the number of instances per class (Fig. 21). This allows us to explore any potential correlation between the number of instances and the observed relative improvement. It can be observed that most classes with higher number of instances provide consistent improvement whereas classes with few instances are rather unpredictable. It underscores the effectiveness of our approach in enhancing instance segmentation performance across a broad range of classes, with only a few exceptions where it falls short.

4.4.2 Exploring the Impact of Object-Centric Prior

In this section, we describe the experiments conducted to evaluate the performance of the approaches mentioned in Section 3.2.2. ie, 1) Using all images of ImageNet 2) Only using object-centric images. We follow the same training procedure as the Experiment 4.4.1. Hence it won't be explained again in this section.

Dataset

We use the ImageNet dataset for training focusing on the subset with ground truth bounding box annotations. We also use the annotated subset of ImageNet, which comprises 38% of the entire dataset. After filtration, we only use 30% of the images employed by the baseline, which is even 5% lesser images compared to Experiment 4.4.1. It is due to removing the multi-instance images from the dataset. We use the same baseline as the Experiment 4.4.1 for comparison and the models are evaluated on all images of COCO validation set.

Results

As it can be observed from Table 3, our method improves in both object detection and instance segmentation tasks. The table shows AP and AP50 of both instance detection and instance segmentation tasks on COCO validation set evaluated using the baseline and our approach. Even by using only 30% of the training data, we could obtain comparable results with the baseline. However, the model trained without overlapping instances (Experiment 4.4.1) outperforms this method by a small margin.

Metric	Baseline		Ours	
	AP	AP50	AP	AP50
bbox	10.74	19.78	**11.32**	**20.43**
segm	8.2	16.62	**8.62**	**17.43**

Table 3: AP and AP50 Bounding Box and Segmentation Evaluation on COCO validation set with baseline and model trained using only object-centric images

4.4.3 Proposed Mask Filtering Method

To address the issue of unwanted background masks included in the pseudo-ground truths, we introduce an enhanced mask filtration approach in Section 3.2.3. This

section details the experimental setup and results associated with our improved mask filtration technique.

Dataset

We use the ImageNet dataset for training and evaluate on COCO, KITTI, PASCAL VOC 2012, Comic and Watercolor datasets. During the initial phase of training, both the baseline and our proposed method utilize the full set of images from ImageNet. However, after the first round of mask filtration in our method, approximately 10% of the images are discarded due to the absence of usable masks. Consequently, the corresponding MaskCut annotations for these excluded images are also removed and the remaining 90% images constitute the training dataset for the self-training stage.

Training Procedure

The baseline follows the training procedure as described in Section 4.3. But our method comes with one extra step of refining MaskCut masks using our modified mask filtration method explained in Section 3.2.3 and training from scratch again. This is followed by self-training as employed in the baseline to improve the performance further. We train both models using 4 and 8 batch sizes to analyze the influence of batch size on performance of the model (Section 4.4.4).

Initial Training Phase: Mask Filtration

Metric	AP	AP50	AR1	AR10
MaskCut	23.3	42.6	41.5	47.5
Our Mask Filtration	**28.3**	**50.3**	**44.4**	**48.0**

Table 4: Quantitative Comparison of CutLER vs Our Proposed Filtration Methods evaluated on annotated ImageNet subset using bounding box labels.

In this section, we evaluate the quality of pseudo-ground truth masks used for training the baseline (MaskCut masks) and our model (Refined MaskCut masks using our Approach 3.2.3).

Table 4 shows a quantitative comparison of the masks used by the baseline and filtered by our approach. Both masks are evaluated on the annotated ImageNet subset using bounding box labels. It is important to note that we only use 15%

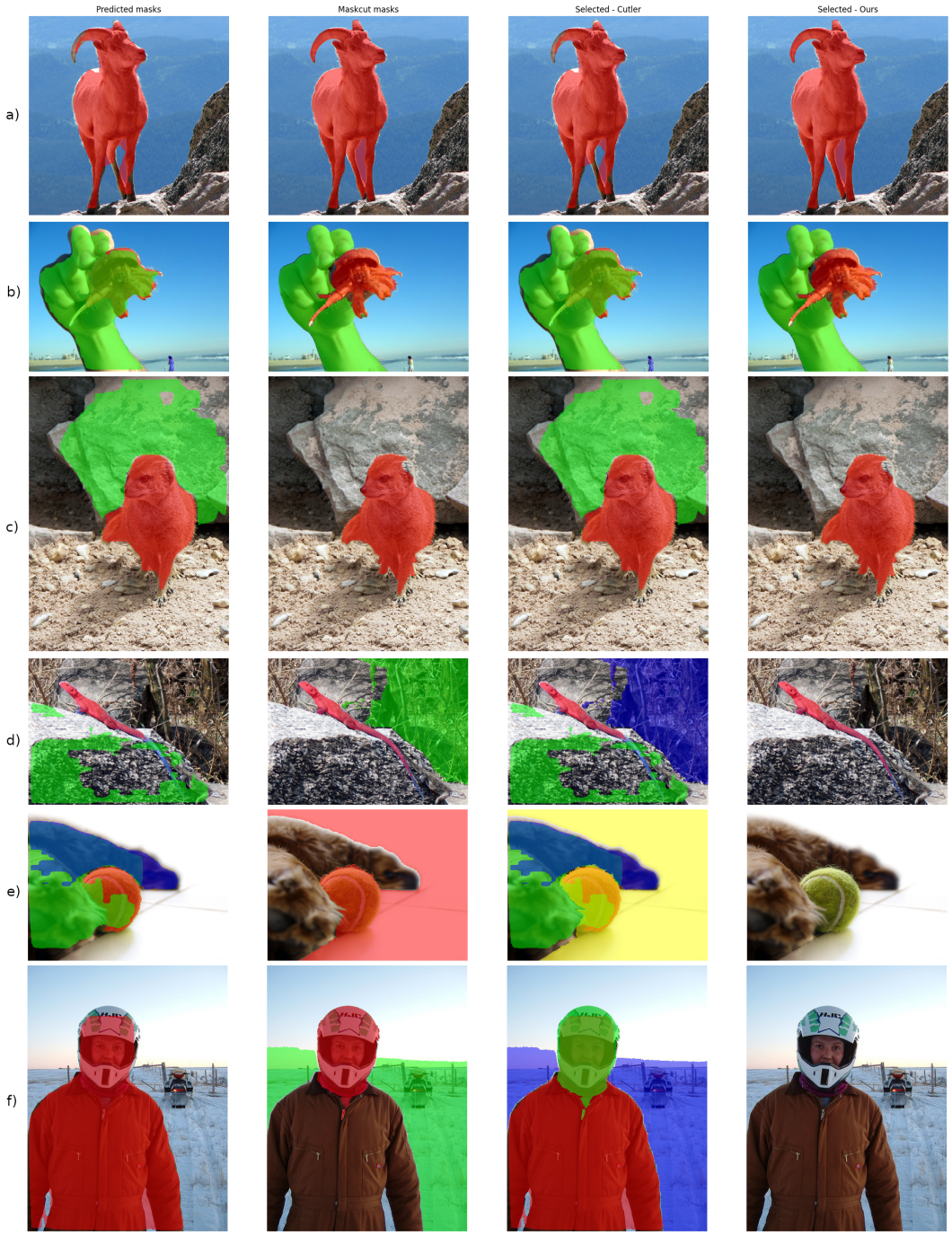

Figure 22: Comparison of Filtered Masks produced by the baseline and our method. Examples illustrates Cutler prediction masks, MaskCut masks, masks selected by CutLER and masks selected by our method respectively (left to right).

less masks compared to MaskCut masks used by the baseline. Still, we outperform MaskCut in all precision and recall metrics.

We observe a significant improvement in precision metrics, driven by the removal of potentially inaccurate masks. Additionally, there is a slight gain in recall, suggesting that our filtration method is balanced enough to produce sufficient masks without excessively limiting coverage of the ground truth. We are using AR10 metric because the number of labels in the ImageNet dataset is consistently fewer than 10 [30] and MaskCut method generates a maximum of 3 masks per image.

Qualitative Examples Figure 22 illustrates the differences between the masks selected by our method and the baseline method across few example images. It is important to note that selected CutLER masks are used for the first stage of self-training in the baseline, whereas we only filter MaskCut masks, which is used for retraining from scratch.

For most images with single instances with distinguishable background, there are no significant differences between masks selected by baseline and our method as given in Fig. 22 (a). Unfortunately a huge part of ImageNet dataset contains single instance object-centric images with simple backgrounds. Hence the improvement that we might obtain can be minimal. Nevertheless, our method works best for examples shown in Fig. 22 (b-d). In Fig. 22 (b-c) our approach successfully removes incorrect masks introduced by CutLER that are overlaid on better MaskCut masks. In Fig. 22 (d), despite both MaskCut and CutLER containing noisy masks, our method effectively filters out these imperfections.

As we discussed earlier in Section 3.2.3, some images are dropped from the training dataset for not having any selected masks. In Fig. 22 (e), the image is dropped due to no selected masks, but CutLER prediction and mask selection are also bad. But in Fig. 22 (f) CutLER selection is certainly better compared to predicted masks. But the image will be dropped in our method. These examples reaffirm that we compromise on quantity for quality in our approach. But nevertheless our method outperforms baseline even with lesser data.

Self-Training Using Proposed Mask Filtration Strategy

All our models use proposed initial mask filtration method. However, using our mask filtration method during self-training results in diminishing evaluation score. The results are illustrated in Table 5. This however was expected, as our masks are already less in number and more refined than baseline.

		COCO	
		AP	AP50
Train	Baseline	11.17	20.12
	Ours	**11.47**	**20.81**
r1	Baseline	**11.70**	**21.15**
	Ours	10.66	19.20

Table 5: AP_{box} **and** $AP50_{box}$ **for Initial Training and Self-Training** evaluated on COCO validation set (Batch size 8) for model trained using our mask filtration during self-training.

On the other hand, using masks generated by our mask filtration method for initial training and baseline mask filtration strategy for self-training yields better results than baseline. All the following experiments follow this combination of filtrations.

Self-Training Using CutLER Mask Filtration Strategy

In this section, we present how using masks generated by our mask filtration method for initial training and baseline mask filtration strategy for self-training outperforms the overall performance of the baseline.

Our evaluations span across five datasets, COCO 2017 Evaluation, KITTI, VOC, Comic, and Watercolor, focusing on instance detection and segmentation tasks. We utilized the precision metric to measure the performance, providing a granular analysis of how our method impacts the accuracy of bounding box and segmentation predictions.

The results summarized in Table 6 demonstrate the performance of both the baseline and our proposed method across different stages of training: the initial training phase and subsequent self-training iterations (r1 and r2). This table provides a comprehensive comparison of the model's effectiveness on various datasets, shedding light on the strengths and weaknesses of each approach throughout the training process.

During the initial training phase, our method outperforms the baseline on the majority of the datasets, including COCO, KITTI, and VOC. Specifically, we observe a notable improvement in precision metrics, indicating that our refined mask filtration process is effective in enhancing the quality of pseudo-ground truths, leading to better detection and segmentation performance. However, our model slightly underperforms on the Comic and Watercolor datasets. After the first round of self-training, our model continues to outperform the baseline on 4 out of 5 datasets. In the second

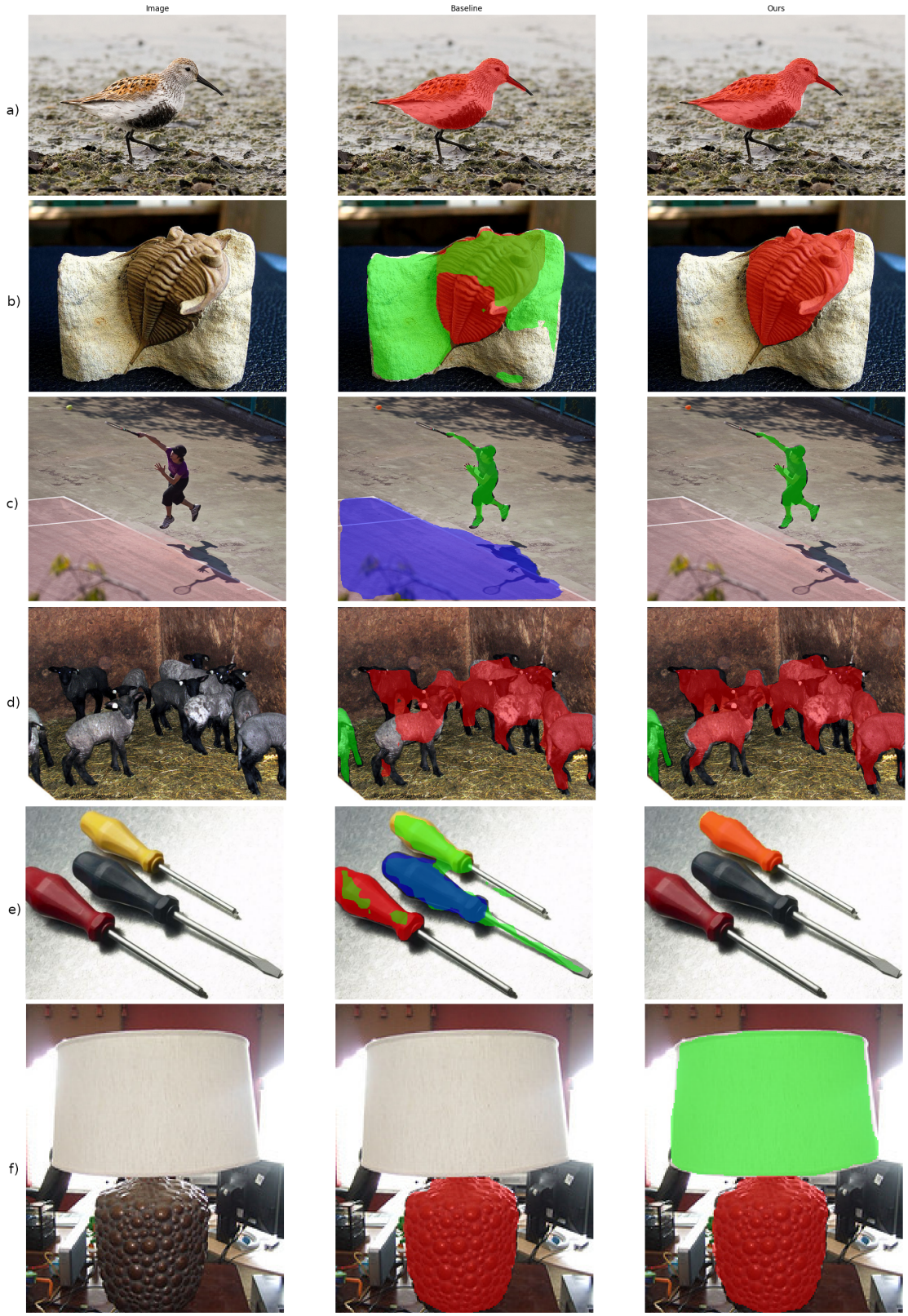

Figure 23: Qualitative Results - Baseline vs Ours Masks predicted by CUtLER and Our model trained with modified mask filtration.

		COCO		KITTI		VOC		Comic		Watercolor	
		AP	AP50	AP	AP50	AP	AP50	AP	AP50	AP	AP50
Train	Baseline	11.17	20.12	4.79	10.27	19.98	36.26	**11.80**	**28.39**	**14.67**	**35.60**
	Ours	**11.47**	**20.81**	**6.28**	**13.48**	**20.24**	**36.56**	10.81	26.50	14.00	35.27
r1	Baseline	11.70	21.15	6.60	14.19	19.56	36.81	**11.05**	**27.53**	12.92	33.45
	Ours	**11.94**	**21.65**	**8.21**	**18.49**	**20.24**	**37.95**	10.80	27.40	**15.37**	**37.74**
r2	Baseline	**11.02**	20.32	6.73	15.02	**18.06**	35.09	**9.90**	**25.36**	**13.59**	**34.31**
	Ours	10.81	**20.47**	**7.69**	**17.04**	17.93	**35.24**	8.94	22.74	12.06	30.02

Table 6: AP_{box} **and** $AP50_{box}$ **for Initial Training and Self-Training** evaluated on COCO validation set (Batch size 8) for model trained using our mask filtration for initial training and baseline mask filtration for self-training.

round of self-training, we observe a slight decline in performance across most datasets for both the baseline and our method, except for minor improvements in the baseline on the KITTI and Watercolor datasets. This decline could be attributed to the model's diminishing returns from further self-training iterations due to more refined masks, especially when using smaller batch sizes, as noted in our experiments in Section 4.4.4.

The evaluation on the Watercolor and Comic datasets has been somewhat inconsistent for both the baseline and our method. We hypothesize that this inconsistency arises from the stark difference between the types of images in these datasets compared to those in ImageNet. While our training predominantly involved real-life images, the Comic and Watercolor datasets consist of stylized, artistic images that pose a different challenge for the models. This disparity in image types likely contributes to the observed inconsistencies in performance.

Qualitative Results Figure 23 presents a qualitative comparison between our method (with combination of mask filtration strategies for self-training) and the baseline. The predictions shown are generated using the most optimized models from both approaches, specifically after the first round of self-training.

Similar to the mask filtration method, for most images with single instances and distinguishable background, there is no significant difference between predicted masks by the baseline and our method as given in Fig. 23 (a). As our method is trained with a small number of filtered masks, our predictions mostly have lesser number of relatively better quality masks (Fig. 23 (b-c)).

Grouping of the instances remains unsolved in both cases (Fig. 23 (d)). However, our method generates more complete masks with finer edges compared to the baseline.

In Fig. 23 (e), our model doesn't predict any masks, but baseline model predicts 4 of them. It could be attributed to the conservative nature of our mask filtration method, predicting less, but higher confidence masks. However, there are few exceptions like Fig. 23 (f) where our model manages to predict more correct masks than the baseline. Even with this limitation, due to the improved quality of the masks, our method outperforms the baseline.

4.4.4 Choice of Batch Size

		Batch size = 8		Batch size = 4	
		AP	AP50	AP	AP50
Train	Baseline	11.17	20.12	10.74	19.78
	Ours	**11.47**	**20.81**	**11.66**	**21.33**
Self-train-r1	Baseline	11.70	21.15	11.35	21.23
	Ours	**11.94**	**21.65**	**11.79**	**21.58**
Self-train-r2	Baseline	**11.02**	20.32	10.51	20.03
	Ours	10.81	**20.47**	**11.25**	**20.20**

Table 7: AP_{box} **and** $AP50_{box}$ **for Initial Training and Self-Training** evaluated on COCO validation set for batch sizes 4 and 8.

In the baseline paper, experiments were conducted using a batch size of 16. However, due to resource limitations, we performed our experiments with smaller batch sizes of 4 and 8. As shown in Table 7, our results indicate a slight improvement in performance with a larger batch size, consistent with expectations that batch size can impact model performance.

Specifically, we observed that for both batch sizes of 4 and 8, the performance of our method and the baseline improved after the first round of self-training (r1) but declined in the second round (r2). This pattern diverges from the findings in the baseline paper, where using a batch size of 16 led to continued performance improvement through the second round of self-training. The observed performance drop in the second self-training round at smaller batch sizes suggests that batch size plays a crucial role in stabilizing the training process, potentially by providing more robust gradient estimates or better generalization.

Furthermore, the influence of batch size on the relative gains achieved through self-training is evident in our experiments. Larger batch sizes tend to offer a more stable training environment, which may explain the continued improvement seen

in the baseline paper with a batch size of 16. Conversely, the smaller batch sizes used in our experiments may introduce greater variance, leading to less consistent performance gains across self-training rounds. This highlights the importance of considering batch size as a key factor in optimizing training pipelines for tasks involving self-training and iterative refinement.

4.4.5 Impact on Recall

		COCO	KITTI	VOC	Comic	Watercolor
Baseline	bbox	**32.06**	25.91	**43.70**	36.97	42.30
	segm	26.16	-	-	-	-
Ours	bbox	31.75	**26.41**	43.34	**38.24**	**43.46**
	segm	**26.44**	-	-	-	-

Table 8: AR_{100} **for the Best Models of the Baseline and Our Method** (Models from the First Self-Training Round) evaluated on COCO validation set for segmentation (segm) and detection (bbox) tasks.

Average Recall (AR) is a valuable metric for unsupervised detection as it does not penalize the models for detecting novel objects which are unlabeled in the dataset. In our case, as our method only produces less number of masks than the baseline, a small reduction in recall was expected. However, it has not affected to the scale we had expected.

Table 8 shows AR_{100} for the best models of our method and the baseline evaluated over 5 different datasets. We observe that our model outperforms the baseline across three datasets, while achieving results that are very close in the remaining two. This suggests that our approach focused on improving precision has had minimal impact on recall.

5 Conclusion and Future Work

In this thesis, we explored and enhanced the CutLER framework for unsupervised object detection and segmentation, focusing on the challenges posed by overlapping instances and the quality of mask annotations. Our investigation began with a detailed analysis of the impact of overlapping instances on model performance. We observed that the presence of overlapping objects in training images often led to confusion in instance differentiation, adversely affecting the model's ability to accurately detect and segment individual objects.

Secondly, by proposing the filtering of non-object-centric images from the training dataset, we aim to support our hypothesis that an object-centric prior is a key contributor to CutLER's state-of-the-art performance. Following the proposed method, we have demonstrated an enhancement in the model's performance.

Building on these findings, we proposed a refined approach to mask filtration aimed at improving the quality of pseudo-ground truth annotations. By selectively filtering out ambiguous and low-certainty masks, we were able to enhance the reliability of the training data, thereby improving the overall performance of the model. This adjustment proved to be particularly beneficial in reducing the inclusion of unwanted background regions.

Through comprehensive experiments across multiple datasets, including COCO, PASCAL VOC, KITTI and others, we demonstrated the effectiveness of our proposed improvements. The results consistently showed that our method outperformed the baseline, particularly in scenarios involving complex and overlapping instances.

In conclusion, this thesis advances the development of unsupervised learning methods by addressing key limitations in the CutLER framework. While the improvements made are notable, they also emphasize the need for further research into more effective techniques for handling complex visual environments.

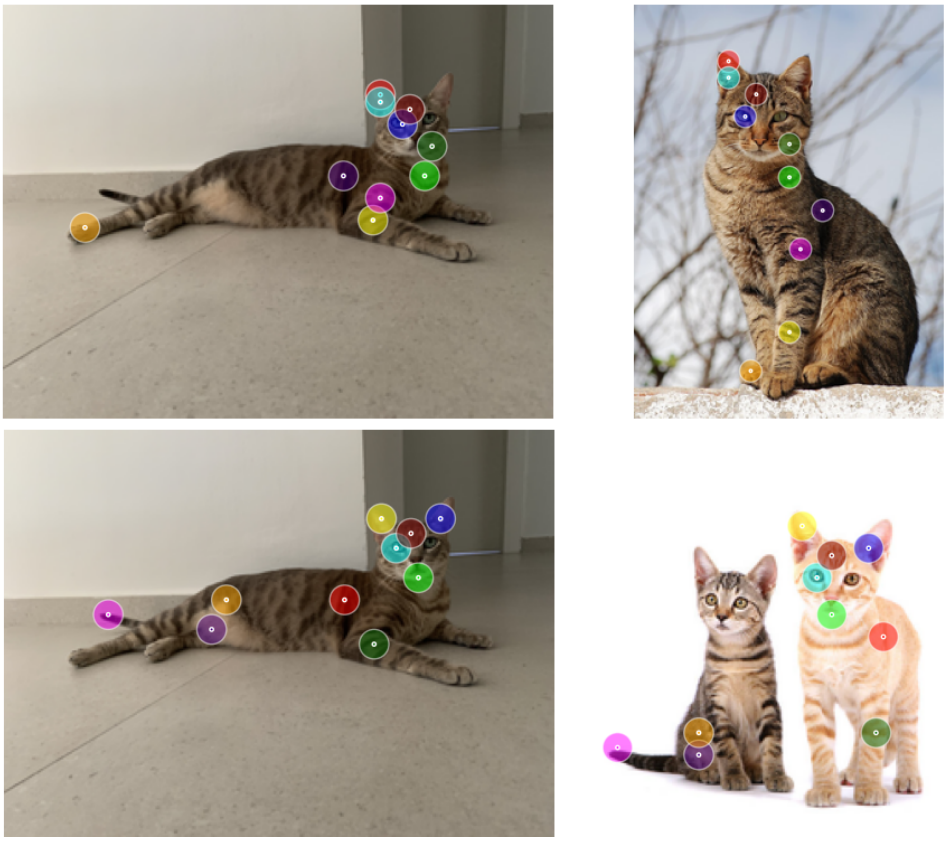

Figure 24: Keypoint Correspondences Using DINO Features. Keypoints are mapped from the left image to the right. Same colored points represent similar features.

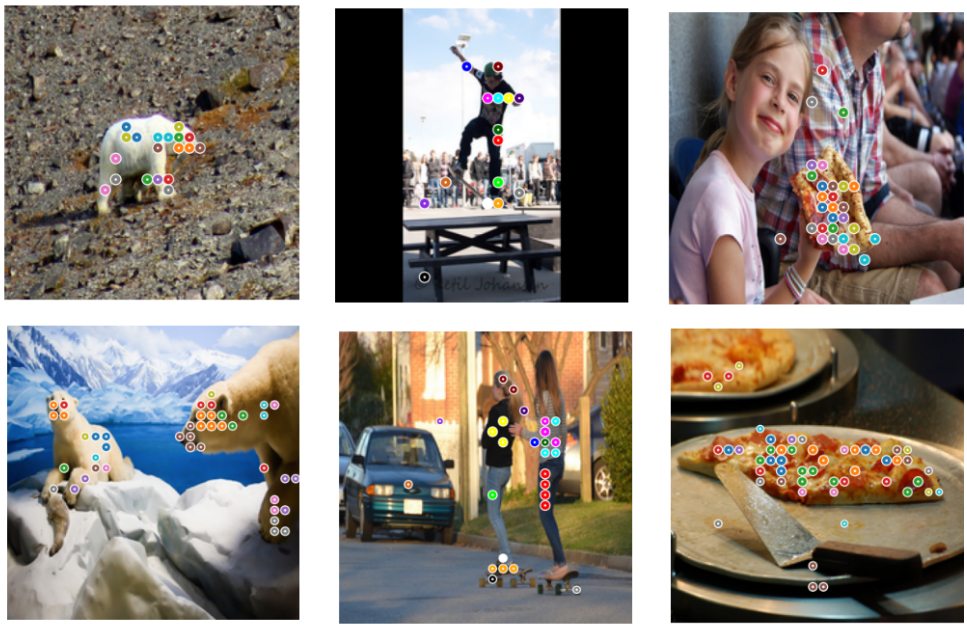

Figure 25: Keypoint Correspondences Using Relaxed Best Buddies. Prototype images at the top and query images at the bottom. Same colored points represent similar features.

5.1 Future Work

5.1.1 Potential of Keypoint Correspondences

For a good part of the thesis work, we worked with keypoint correspondences extracted from ViT-DINO model to extract semantically similar parts in the image and use this information to separate the instances. The motivation was to implement a method for distinguishing overlapping instances in an image and potentially devise a semi-supervised method (using few prototypes for each class) to train an object detection model. We briefly describe our attempted approach in this section without going into too much detail.

Best Buddies

The correspondences are calculated using Best Buddy algorithm [41]. A pair of patches is considered a "Best Buddy" if the patch in the first image chooses a patch in the second image as its best match, and that same patch in the second image also chooses the original patch in the first image as its best match. We apply the Best Buddies algorithm to the similarity matrix derived from the key descriptors

retrieved from the final layer of the DINO-ViT model for both images, generating the correspondences.

The fundamental problem of point correspondences involving multiple instances is illustrated in Fig. 24. In the first image pair, while the correspondences are accurate, they fail to cover a significant portion of the instances. In the second pair, the correspondences are scattered between two instances. This indicates either an insufficient number of correspondences or that the correspondences are unevenly divided between the instances, preventing clear distinction.

Relaxed Best Buddies

To generate sufficient number of correspondences for each instances in a multi-instance image, we "relax" the Best Buddy algorithm by allowing top N best matches for each patch instead of choosing just one. In doing so, we increase the number of correspondences, but slightly compromise the quality of those correspondences.

Figure 25 shows the keypoint correspondences between the prototype and query images generated using our relaxed algorithm. We can see small clusters of points representing similar features in both instances. We explored our options to geometrically separate the instances using the semantic information in the correspondences using graph-cut approaches [5, 42]. But we were unable to find a promising approach or inspiration suited for the problem from our research. For instance, Integer programming for multidimensional assignment problem [43] has a strict initial graph specification which restricts us to adapt it to our problem.

By considering the limitations of implementing a semi-supervised method using keypoint correspondences, we focused mostly on improving the unsupervised instance detection and segmentation by diving into the current state-of-the-art method CutLER. Nevertheless, we acknowledge the potential of utilizing keypoint correspondences for instance detection and believe this approach warrants further exploration. It could also potentially lead to a more effective method for separating overlapping instances in future research.

5.1.2 Unsupervised Detection of Overlapping Instances

Based on our findings, a key area for future research could be the development of unsupervised methods designed to better detect both individual and overlapping instances within images. It would be beneficial to extend our approach beyond merely filtering out images with overlapping instances. Developing techniques to further separate and identify individual instances within these overlapping regions

could address a significant challenge in unsupervised learning. Given that many unsupervised methods currently struggle with distinguishing overlapping instances, this direction holds promise for enhancing the accuracy and robustness of instance detection and segmentation in complex scenarios.

6 Acknowledgments

I extend my sincere gratitude to Dr. Thomas Brox for his invaluable contribution as the examiner of this research thesis. I also wish to thank Dr. Matthias Teschner for graciously agreeing to be my second examiner and for showing a keen interest in my work.

My heartfelt thanks go to my supervisor, Silvio Galesso, whose unwavering support and mentorship have been the cornerstone of this research. Silvio's continuous encouragement, constructive feedback, and expert guidance have been invaluable throughout the entire research process.

I would also like to acknowledge the support of my friends, whose encouragement and understanding have been a source of motivation. Additionally, I am grateful to Sarah and ChatGPT for supporting me with the difficult writing process by covering my underwhelming English language skills.

The completion of this thesis has been a significant personal achievement, and I am genuinely appreciative of the contributions and support from these individuals and resources. Their influence has been instrumental in reaching this milestone.

Bibliography

[1] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[2] S. Amir, Y. Gandelsman, S. Bagon, and T. Dekel, "Deep vit features as dense visual descriptors," *arXiv preprint arXiv:2112.05814*, vol. 2, no. 3, p. 4, 2021.

[3] X. Wang, R. Girdhar, S. X. Yu, and I. Misra, "Cut and learn for unsupervised object detection and instance segmentation," 2023.

[4] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9650–9660, 2021.

[5] Y. Wang, X. Shen, Y. Yuan, Y. Du, M. Li, S. X. Hu, J. L. Crowley, and D. Vaufreydaz, "Tokencut: Segmenting objects in images and videos with self-supervised transformer and normalized cut," *arXiv preprint arXiv:2209.00383*, 2022.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[7] X. Wang, Z. Yu, S. D. Mello, J. Kautz, A. Anandkumar, C. Shen, and J. M. Alvarez, "Freesolo: Learning to segment objects without annotations," 2022.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[9] L. Weng, "The transformer family," *lilianweng.github.io*, Apr 2020.

[10] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," 2023.

[11] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," 2020.

[12] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," 2020.

[13] Y. M. Asano, C. Rupprecht, and A. Vedaldi, "Self-labelling via simultaneous clustering and representation learning," 2020.

[14] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, "Unsupervised learning of visual features by contrasting cluster assignments," 2021.

[15] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko, "Bootstrap your own latent: A new approach to self-supervised learning," 2020.

[16] P. Engstler, L. Melas-Kyriazi, C. Rupprecht, and I. Laina, "Understanding self-supervised features for learning unsupervised instance segmentation," 2023.

[17] O. Siméoni, G. Puy, H. V. Vo, S. Roburin, S. Gidaris, A. Bursuc, P. Pérez, R. Marlet, and J. Ponce, "Localizing objects with self-supervised transformers and no labels," 2021.

[18] W. V. Gansbeke, S. Vandenhende, and L. V. Gool, "Discovering object masks with transformers for unsupervised semantic segmentation," 2022.

[19] T. Ishtiak, Q. En, and Y. Guo, "Exemplar-freesolo: Enhancing unsupervised instance segmentation with exemplars," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15424–15433, June 2023.

[20] G. Ghiasi, Y. Cui, A. Srinivas, R. Qian, T.-Y. Lin, E. D. Cubuk, Q. V. Le, and B. Zoph, "Simple copy-paste is a strong data augmentation method for instance segmentation," 2021.

[21] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[22] O. Siméoni, G. Puy, H. V. Vo, S. Roburin, S. Gidaris, A. Bursuc, P. Pérez, R. Marlet, and J. Ponce, "Localizing objects with self-supervised transformers and no labels," 2021.

[23] W. V. Gansbeke, S. Vandenhende, and L. V. Gool, "Discovering object masks with transformers for unsupervised semantic segmentation," 2022.

[24] Y. Wang, X. Shen, Y. Yuan, Y. Du, M. Li, S. X. Hu, J. L. Crowley, and D. Vaufreydaz, "Tokencut: Segmenting objects in images and videos with self-supervised transformer and normalized cut," 2023.

[25] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," 2021.

[26] S. Maji, N. K. Vishnoi, and J. Malik, "Biased normalized cuts," in *CVPR 2011*, pp. 2057–2064, 2011.

[27] H. V. Vo, P. Pérez, and J. Ponce, "Toward unsupervised, multi-object discovery in large-scale image collections," in *Computer Vision – ECCV 2020* (A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, eds.), (Cham), pp. 779–795, Springer International Publishing, 2020.

[28] Z. Cai and N. Vasconcelos, "Cascade r-cnn: High quality object detection and instance segmentation," 2019.

[29] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," 2018.

[30] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[31] M. Tang, A. Djelouah, F. Perazzi, Y. Boykov, and C. Schroers, "Normalized cut loss for weakly-supervised cnn segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[32] A. Ziegler and Y. M. Asano, "Self-supervised learning of object parts for semantic segmentation," 2022.

[33] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, "Self-training with noisy student improves imagenet classification," 2020.

[34] Y. Wang, X. Shen, S. X. Hu, Y. Yuan, J. L. Crowley, and D. Vaufreydaz, "Self-supervised transformers for unsupervised object discovery using normalized cut," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14543–14553, June 2022.

[35] E. Gasparim, F. Valencia, and C. Varea, "Invariant generalized complex geometry on maximal flag manifolds and their moduli," *Journal of Geometry and Physics*, vol. 163, p. 104108, May 2021.

[36] O. Russakovsky, Y. Lin, K. Yu, and L. Fei-Fei, "Object-centric spatial pooling for image classification," in *Computer Vision – ECCV 2012* (A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, eds.), (Berlin, Heidelberg), pp. 1–15, Springer Berlin Heidelberg, 2012.

[37] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," 2015.

[38] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[39] N. Inoue, R. Furuta, T. Yamasaki, and K. Aizawa, "Cross-domain weakly-supervised object detection through progressive domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[40] C. Sutton and A. McCallum, "An introduction to conditional random fields," 2010.

[41] K. Aberman, J. Liao, M. Shi, D. Lischinski, B. Chen, and D. Cohen-Or, "Neural best-buddies: sparse cross-domain correspondence," *ACM Transactions on Graphics*, vol. 37, p. 1–14, July 2018.

[42] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superglue: Learning feature matching with graph neural networks," 2020.

[43] J. L. Walteros, C. Vogiatzis, E. L. Pasiliao, and P. M. Pardalos, "Integer programming models for the multidimensional assignment problem with star costs," *European Journal of Operational Research*, vol. 235, no. 3, pp. 553–568, 2014.