

Master's Thesis

Exploring Possible Improvements for Unsupervised Instance 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. 2023 – 23. 08. 2023

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 two 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 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

1	Introduction	1
1.1	Motivation	1
1.2	Overview of CutLER	1
1.3	Contribution and Key Insights	3
1.4	Outline	3
2	Background and Related Works	5
2.1	Vision Transformer	5
2.1.1	Patch Tokens and Positional Encoding	5
2.1.2	Transformer Encoder	6
2.2	Self-Supervised Feature Learning	8
2.2.1	Contrastive Learning	8
2.2.2	Clustering-Based Feature Learning	9
2.2.3	Distillation-Based Methods	9
2.3	DINO	10
2.3.1	Knowledge Distillation	10
2.3.2	Training Process	10
2.4	Unsupervised Object Detection and Instance Segmentation	11
2.5	CutLER	12
2.5.1	Normalized Cut	13
2.5.2	TokenCut	14
2.5.3	MaskCut	14
2.5.4	DropLoss	16
2.5.5	Copy-Paste Augmentation	16
2.5.6	Training	17
2.5.7	Mask Refinement in CutLER	17
3	Approach	19
3.1	Accessing the limitations of CutLER	19
3.1.1	Dependence on Initial Masks	19

3.1.2	Overlapping Instances	20
3.1.3	Images with Complex Background	24
3.2	Mask Filtration	24
3.2.1	Baseline Mask Filtration Method	25
3.2.2	Proposed Mask Filtering Method	28
4	Experiments and Results	33
4.1	Datasets	33
4.1.1	ImageNet	33
4.1.2	COCO	34
4.1.3	PASCAL VOC	34
4.1.4	KITTI	34
4.1.5	Comic and Watercolor	34
4.2	Metrics	35
4.2.1	Average Precision	35
4.3	Implementation Details	36
4.3.1	Training Data	36
4.3.2	MaskCut	36
4.3.3	Detector	36
4.3.4	Self Training	37
4.3.5	Resources	37
4.4	Experiments	37
4.4.1	Exploring the Impact of Overlapping Instances	37
4.4.2	Exploring the Impact of Object-Centric Prior	42
4.4.3	Proposed Mask Filtering Method	43
4.4.4	Choice of Batch Size	48
4.4.5	Recall	49
5	Conclusion and Future Work	51
5.1	Future Work	52
5.1.1	Keypoint Correspondences	52
5.1.2	Sensitivity to Hyperparameters	53
5.1.3	Unsupervised Detection of Overlapping Instances	53
6	Acknowledgments	55
Bibliography		56

List of Figures

1	Comparison of MaskCut and CutLER Outputs	2
2	ViT Architecture	6
3	Transformer Encoder Architecture	7
4	DINO Architecture	11
5	CutLER Overview	13
6	MaskCut Flow	15
7	Copy-Paste Augmentation	16
8	Dependence on Initial Masks	20
9	Cutler’s Performance on Images with Overlapping Instances	21
10	MaskCut vs Part Co-Segmentation Masks	21
11	Cutler’s Performance on Images with Complex Background	25
12	Baseline Training Pipeline	26
13	Mask Filtration in Baseline	26
14	Mask Filtration in Baseline	27
15	Proposed Training Pipeline	29
16	Mask Filtration in Proposed Method	29
17	Mask Filtration in Proposed Method in Self-Training	31
18	Training without Overlapping Instances - Class-wise Comparison	40
19	Relative Improvement against Number of Instances	41
20	Mask Filtration Outputs - Baseline vs Ours	44
21	Qualitative Results - Baseline vs Ours	47
22	Keypoint correspondences using relaxed best buddies	52

List of Tables

1	Evaluation of Models Trained with and without Overlapping Instances	39
2	Evaluation of Models Trained with and without Overlapping Instances with Evaluation Dataset Split	39
3	Evaluation of baseline vs Model Trained with only object-centric images	42
4	AP_{box} and $AP50_{box}$ for Training and First Self-Training Round	45
5	AP_{box} and $AP50_{box}$ for Training and Self-Training	46
6	AP_{box} and $AP50_{box}$ for Training and Self-Training for Different Batch Sizes	49
7	AR_{100} for the Best Models of Baseline and Our Method	49

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 such as grouping of nearby 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

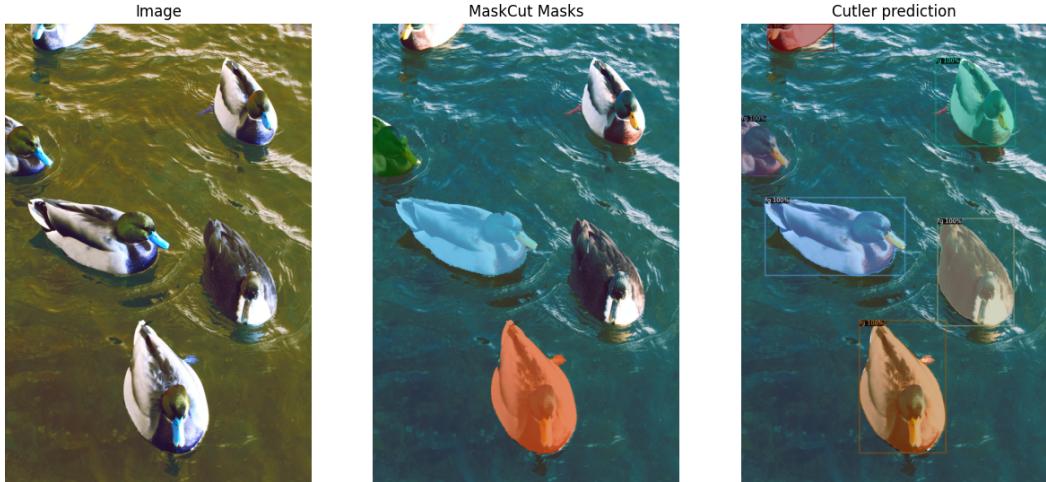


Figure 1: Maskcut and CutLER outputs. Figure illustrates the outputs of MaskCut and CutLER with N=3 (Number of masks generated).

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 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 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, which uses a ResNet101 backbone. Specifically, CutLER improves the average precision (AP50) 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. **Refining maskcut masks:** On top of self training, we refine mask-cut masks using CutLER predictions to remove noisy masks and retraining to yield better performance.

1.4 Outline

- **Chapter 2:** Explores the foundational concepts of Vision Transformers and attention mechanisms, with a detailed introduction to DINO features. Also covers the CutLER training pipeline and methods for mask refinement.
- **Chapter ??:** Explores recent advances in self-supervised feature learning and developments in unsupervised object detection and semantic segmentation.
- **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.
- **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 NLP. The highlight of the paper is reusing the transformer encoder from the revolutionary work Vaswani et al. [6] 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 sequence of words or tokens in [6], 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.

$$z^0_i = z_i + p_i \quad (1)$$

For each token, instead of using sinusoidal position encodings [6] 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.

$$Z = [z_{class}; z^0_1; z^0_2; \dots; z^0_N] \quad (2)$$

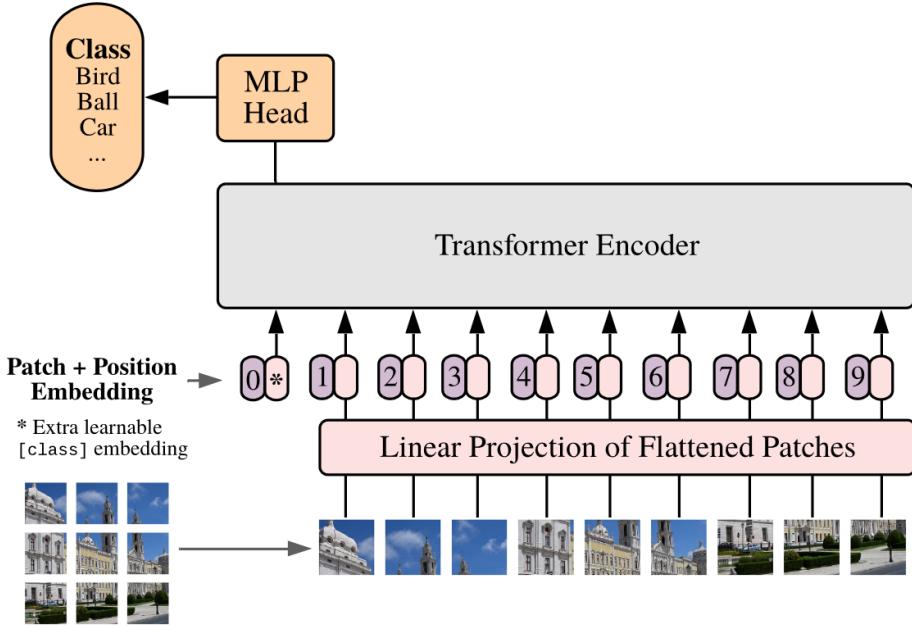


Figure 2: ViT Architecture. ViT architecture from [1].

Apart from [6], 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 look 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 [7] of individual attention outputs constitute the final output. Figure 3 illustrates the architecture of the transformer encoder. We briefly look into each part.

Multi-Head Self-Attention (MSA)

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

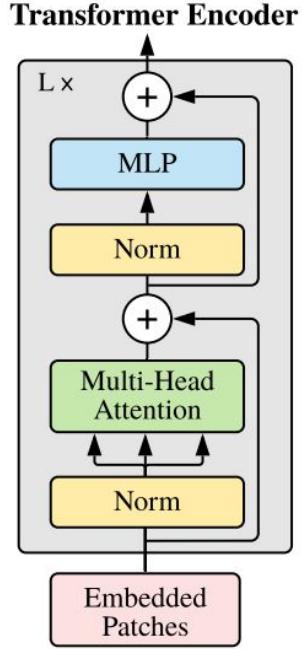


Figure 3: Transformer Encoder Architecture. Illustration of the Transformer encoder architecture in ViT [1].

$$\text{Queries } Q = zW_i^Q \quad (3a)$$

$$\text{Keys } K = zW_i^K \quad (3b)$$

$$\text{Values } V = zW_i^V \quad (3c)$$

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 ,

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

The outputs from all heads are concatenated and linearly transformed. Given $W^O \in \mathbb{R}^{h-d_k \times D}$:

$$MSA(z) = \text{Concat}(\text{head}_i, \text{head}_2, \dots, \text{head}_h)W^O \quad (5)$$

Layer Normalization and Residual Connections

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

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

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

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

$$\text{MLP}(x) = W_2(\text{GELU}(W_1x)) \quad (8)$$

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 $W_{\text{class}} \in \mathbb{R}^{C \times D}$:

$$\text{logits} = W_{\text{class}} \cdot z''_{\text{class}} \quad (9)$$

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 [8]. 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) [9] 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 [10] can capture multi-scale structures and found to be effective for diverse datasets. But found to be computationally expensive and needs careful tuning of the self-supervised task. SwAV [11] 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. BYOL (Bootstrap Your Own Latent) [12] 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 ConvNets, 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 typically 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.

$$\theta_t \leftarrow m\theta_t + (1 - m)\theta_s \quad (10)$$

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

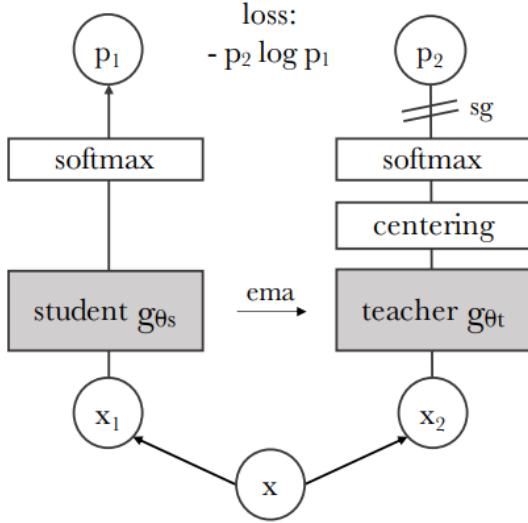


Figure 4: Architecture of DINO Illustration provided in [4].

from both models are projected into a lower-dimensional space using projection heads. 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 4

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

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 Vision Transformer (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 is superior to the existing methods if the image contains only one instance. The superiority of DINO features to separate foreground

and background has been affirmed by later works [13]. Building on this observation, both LOST [14] 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. But it can't be used as a pre-trained model for down stream tasks. But CutLER [3] not only can detect multiple instances, the model can be further used as a pretrained model for label-efficient and fully-supervised learning.

MaskDistill [15] advocate 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). Then a final segmentation model is trained using the found object masks. Even though MaskDistill produces similar quality masks compared to MaskCut, as it only produces one class agnostic mask per image and MaskCut produces up to N number of masks per image to use as pseudo labels, MaskCut weighs over MaskDistill in quantity.

FreeSOLO [16] and the follow up work Exemplar-FreeSOLO [17] (with its addition of a randomly drawn pool of exemplars used in a contrastive learning loss) generates coarse segmentation masks with low quality and refines it further through self training similar to CutLER. But the poor quality of the coarse maps is a major draw back of this method, where as CutLER masks made by the MaskCut [3, 5] algorithm are usually better in quality and quantity than the initial masks used by MaskDistill [15] and FreeSOLO [16].

As CutLER outperforms other methods in most cases, including producing better pseudo ground truth masks, ability to detect multiple instances, compatibility with various detection architectures, usable as pretrained model for supervised detection, our work would mostly focus on studying and improving the performance of 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 [18] 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.

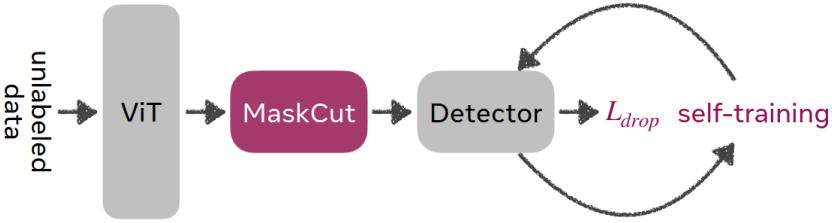


Figure 5: CutLER overview The flow consists of MaskCut for extracting coarse masks from the features of a self-supervised ViT. Following this, a detector utilizing a loss dropping strategy designed to be resilient against objects that MaskCut may overlook is used. Additionally, the model undergoes further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

2.5.1 Normalized Cut

Normalized Cut (NCut) [19] 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. A basic Cut is defined as the sum of the edge weights that are severed by the partition as given in Eq. 12:

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

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:

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

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.

$$(D - W)x = \lambda Dx \quad (14)$$

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 be 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.

2.5.3 MaskCut

Like TokenCut [5], MaskCut considers image segmentation problem as a graph partitioning task [19]. 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 based on the similarity of patches

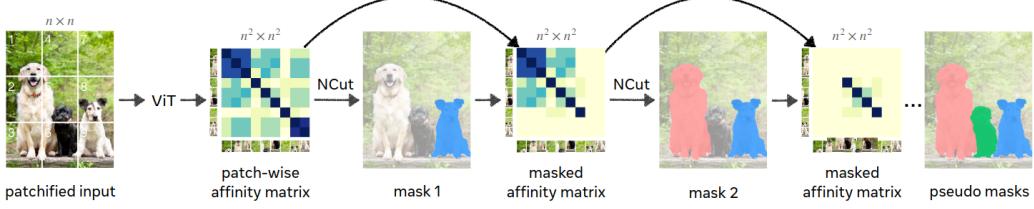


Figure 6: **MaskCut** works on the patch-wise similarity matrix for the image using a self-supervised DINO [4] model feature. N=3 defines the number of times NCut [19] is repeated on the background. In this case, 3 instances will be discovered per image in each step. Illustration taken from [3]

in the DINO feature space. Following recent methods [20, 21, 22], they specifically employ the cosine similarity of 'key' features from the final attention layer of the DINO-pretrained model, represented as:

$$W_{ij} = \frac{K_i \cdot K_j}{\|K_i\|_2 \|K_j\|_2} \quad (15)$$

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 (Defines the number of times NCut is repeated. In this case, 3 instances will be discovered per image) Building on the work of [22, 23], a patch-wise similarity matrix for the image using features from a self-supervised DINO model [4] is created. Normalized Cuts [19] 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 [24] 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 [23, 25], we assert that foreground mask should contain the patch corresponding to the maximum absolute value in the second smallest eigenvector. If condition 1 is not satisfied and current foreground contains two corners, background and foreground are switched.



Figure 7: Illustration of Copy-Paste augmentation from [18]

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.

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

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 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 [18] 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

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[26] yields better results compared to Mask R-CNN [27].

First, pseudo-ground truth masks are generated using MaskCut for all images in Imagenet [28] training set. The detector with a ResNet-50 [29] 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. The CutLER mask predictions of each image with confidence score > 0.7 generated using the final model of last training phase and corresponding MaskCut masks which doesn't overlap more than 50% with the predicted masks together forms the pseudo-ground truth masks for the first round of self-training. 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 is used to compare with predicted masks. Each self training round is consist of 80K rounds and does not use DropLoss as we obtain comparatively good quality masks in the first round it self. The detailed information about the training are explained later in Section 4.3.

2.5.7 Mask Refinement 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.

3 Approach

In the field of unsupervised instance detection and segmentation, CutLER [3] gives a strong performance by exploiting a object-centric prior by training on ImageNet [28], 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 it’s 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 Accessing the limitations of CutLER

Even though CutLER is the state-of-the-art model for unsupervised instance detection and segmentation, it still has several drawbacks. Mainly 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 8: 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. 8 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 3.2.2.

3.1.2 Overlapping Instances

Identifying instances using an unsupervised instance detection or segmentation method presents significant challenges, especially when instances are closely positioned or overlapping in an image. In such scenarios, the algorithm must discern subtle differences in texture, color, and shape without the benefit of labeled training data. Overlapping objects often blend together, making it difficult for the model to accurately segment and differentiate them as distinct entities. This lack of explicit supervision complicates the model’s ability to learn and generalize the spatial relationships and boundaries between objects.

As illustrated in Fig. 9, in almost all cases of images with overlapping instances, MaskCut and CutLER groups those instances together. MaskCut primarily separates the foreground from the background rather than differentiating between individual instances [13]. As a result, it can successfully detect the fish but groups the people together as there is an overlap between them. Consequently, in images with overlapping instances, both MaskCut and CutLER generate outputs that are more akin to

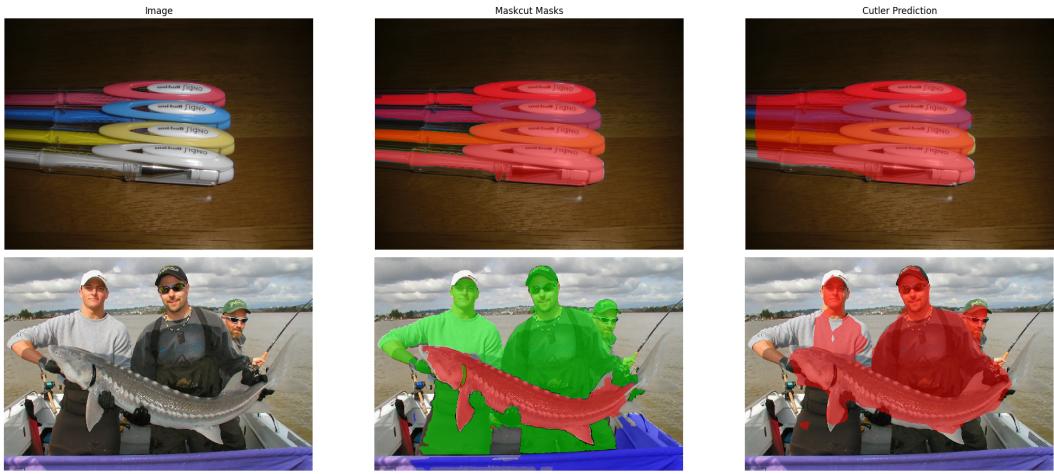


Figure 9: Grouping of overlapping instances in MaskCut and CutLER outputs, which is a common problem in most of the existing unsupervised instance detection methods.

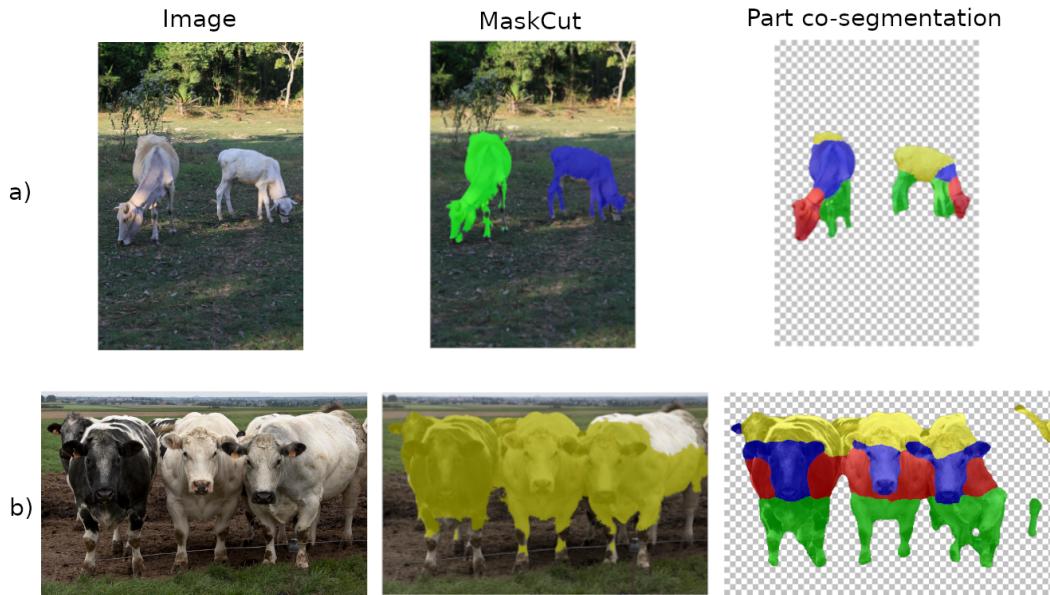


Figure 10: MaskCut vs Part Co-Segmentation Masks. Discriminative features that could help differentiate instances are present in DINO features, as evidenced in n-part co-segmentation. But these features are not leveraged by MaskCut.

semantic segmentation than true instance segmentation.

MaskCut is also not effectively making use of the discriminative DINO features that could help differentiate the instances. As it can be observed from Fig. 10, 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.

The problem is not exclusive to MaskCut or CutLER, but most of the existing methods [13, 25, 30] also address 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 an graph-cut algorithm that fits 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 next section.

Approach to Estimate the Impact of Overlapping Instances

Based on the observation that MaskCut often groups overlapping instances into a single mask, 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 is 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 $\tau > 10\%$ is taken as the criteria to filter images for approach 2 and 3. As ImageNet contains mostly object-centric single instance images, Third approach (only using images with overlapping instances) would have significantly less 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.

Estimating 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 [13, 31].

Object-centricity is hard to define. As we couldn't find a standard way to define object-centricity for our research [32, 31], 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.

3.1.3 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 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, which is explained in detail in section 3.2.1.

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.

3.2 Mask Filtration

Generating initial pseudo-ground truth masks using a pre-trained model or some heuristic methods may contain errors or inaccuracies. The presence of incorrect



Figure 11: Images with complex backgrounds impacting MaskCut and CutLER outputs, leading to undesired background mask generation

masks 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. [33] and Wang et al. [34] explore these approaches. Iterative refinement helps in progressively reducing this noise, leading to cleaner and more reliable labels [35]. 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.

3.2.1 Baseline Mask Filtration Method

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. 12, 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.

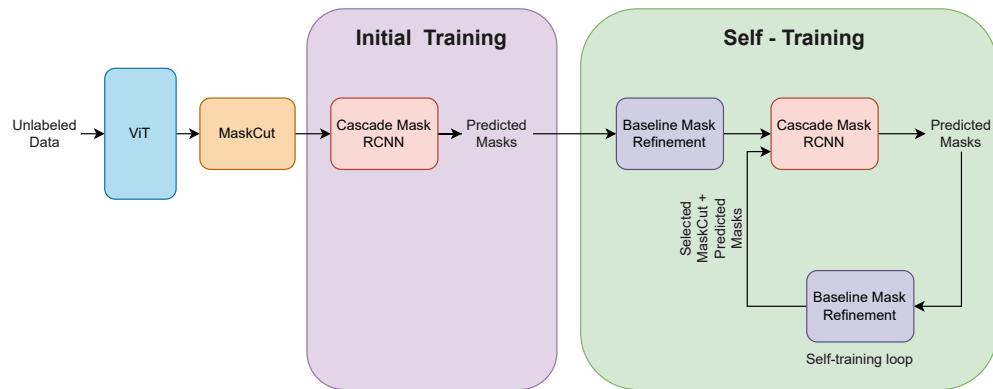


Figure 12: Baseline training pipeline with repeated mask filtration and self-training

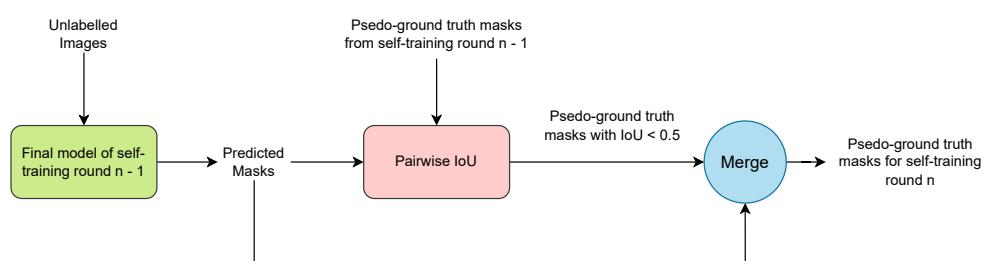


Figure 13: Mask Filtration in Baseline before each self-training loop

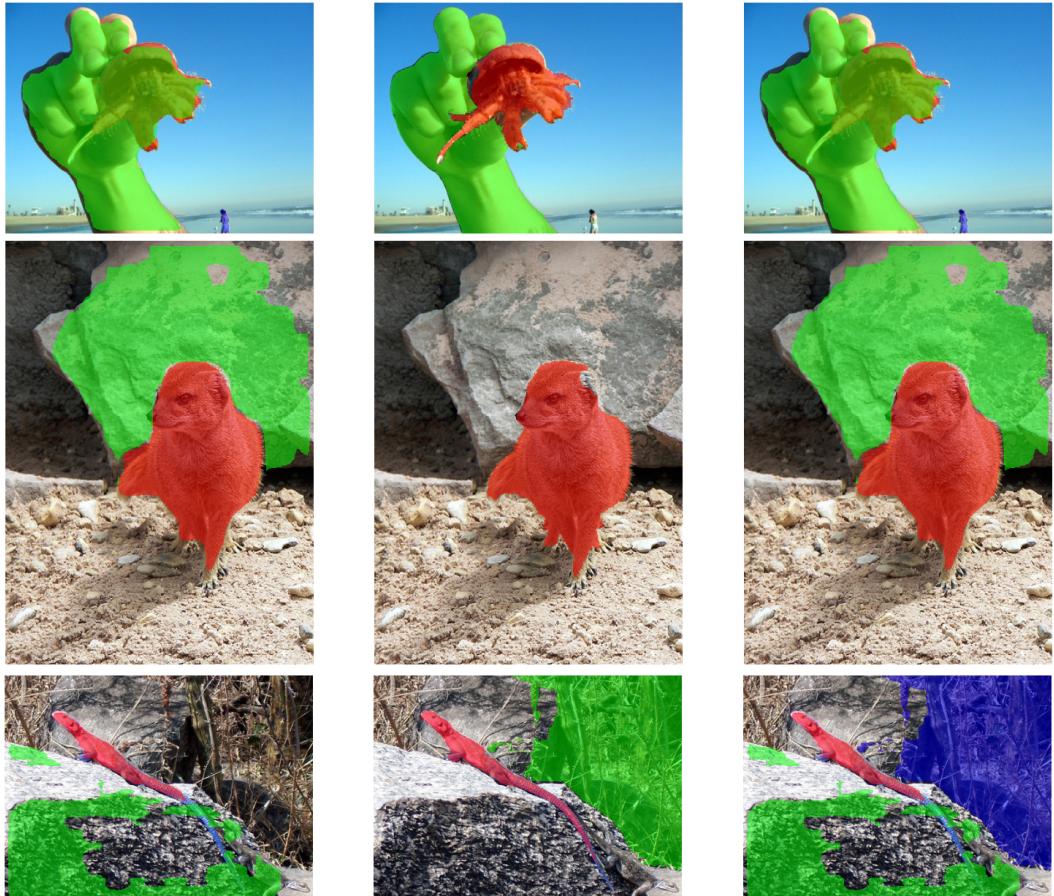


Figure 14: Mask Filtration in Baseline. Examples illustrates CutLER prediction masks, MaskCut masks and masks selected by CutLER mask filtration method (left to right)

Mask Filtration and Self-Training

Baseline mask filtration process is illustrated in Fig. 13. 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 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 may introduce 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 upto 3 self-training loops by the authors. We will be running 2 self-training loops for both the baseline and proposed method.

Qualitative Examples

Figure 14 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 doesn't overlap much along with CutLER prediction masks, which adversely affect the evaluation score. Our method intents to remove these masks to generate better quality pseudo-ground truths.

3.2.2 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.

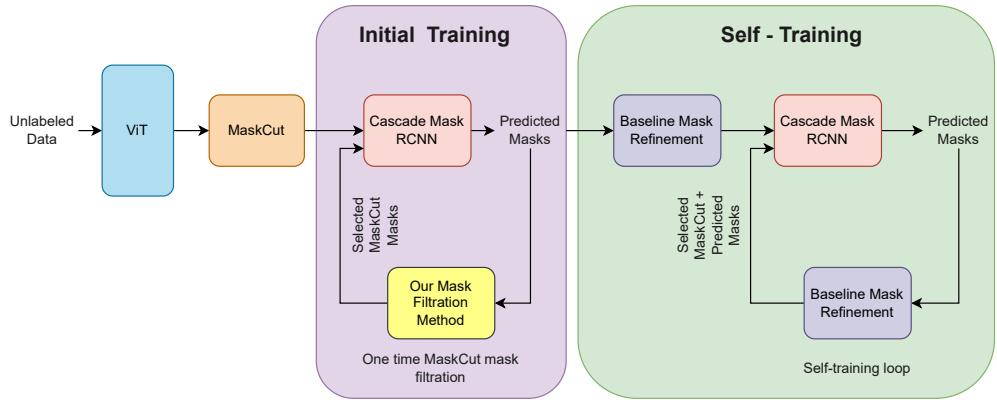


Figure 15: Proposed training pipeline featuring a one-time MaskCut mask filtration followed by multiple self-training loops with our mask filtration method.

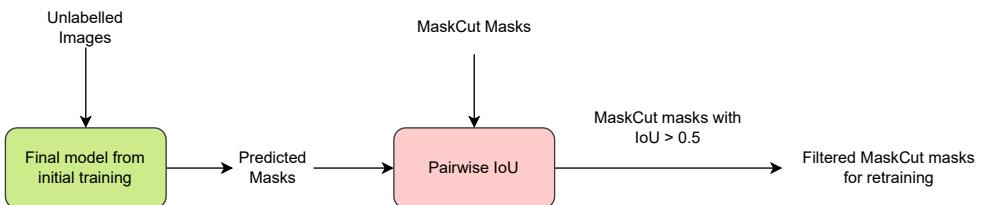


Figure 16: Mask Filtration in Our Method, used to filter MaskCut masks after the initial training.

Initial Training

As illustrated in Fig. 15, following the initial training phase identical to the baseline (160K iterations using Copy-Paste augmentations and DropLoss), we introduce an additional step to refine MaskCut masks. This refinement involves preserving only high-certainty masks by comparing them against CutLER predictions.

Proposed Mask Filtration and Retraining

Proposed mask filtration method is illustrated in Fig. 16. 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 $\text{IoU} < 0.5$ from the batch IoU matrix, we choose MaskCut masks that correspond to mask pairs with an $\text{IoU} > 0.5$.

The idea is to filter highly certain MaskCut masks and discard possible incorrect masks. This selected MaskCut masks are treated as the new pseudo-ground truth and we train from scratch for 160K iterations. 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.

Once the filtered MaskCut masks are obtained, we train the detector from scratch again for 160K iterations with Copy-Paste augmentations and DropLoss. We are effectively repeating the same initial training process, but with better masks.

Further Mask Filtration and Self-Training

For self-training, we follow training pipeline of the baseline, training for 80K iterations are without using DropLoss. We use our mask filtration method illustrated in Fig 17 to filter masks between self-training rounds.

The only modification to the baseline approach is selecting pseudo-ground truth masks with $\text{IoU} > 0.5$ with predicted masks, rather than those with $\text{IoU} < 0.5$. This way, we always retain masks with high certainty. But this also restricts the exploration. Limiting pseudo-ground truth labels to only high-confidence masks

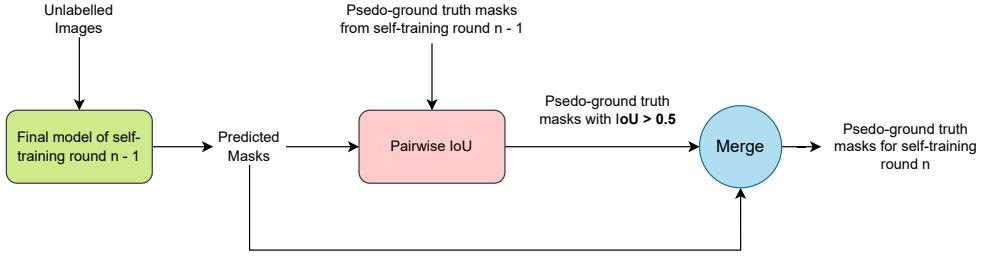


Figure 17: Mask Filtration in Proposed Method, used to filter masks during self-training.

could constrain improvements in the self-training round.

As we are using lesser number of more accurate masks, we expect an improvement of precision. However, the recall could decrease by a small factor if there are not enough masks generated with respect to the ground truth masks. 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 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 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-refinement process(Images with no annotations are removed).

4.1.2 COCO

The COCO (Common Objects in Context) dataset [36] 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 segmentation annotations as ground truths.

4.1.4 KITTI

The KITTI dataset [37] 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 [38], 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, as in our approach for the KITTI dataset.

4.2 Metrics

We mostly use precision metrics to evaluate performance of the models in our work. This include 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 well 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.

$$P = \frac{TP}{TP + FP} \quad (17)$$

$$R = \frac{TP}{TP + FN} \quad (18)$$

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 and P_n are the recall and precision at the n^{th} threshold.

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

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

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}} \quad (20)$$

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.

4.3 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 training including 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 in section 4.4.1.

4.3.2 MaskCut

We apply MaskCut with N=3, generating upto 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) 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 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 each stage, along with CutLER mask predictions with confidence score > 0.7 generated using the model from previous stage, Maskcut masks which have $\text{IoU} < 0.5$ with the CutLER prediction masks together make the pseudo ground truth masks for that stage. 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.

4.3.5 Resources

Generating MaskCut annotations for all images in ImageNet is supposed to 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.

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 3.1.2. 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 the 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, entire ImageNet dataset is used (around 1.3 million images) and doesn't use any bbox annotations. For our approach, we utilize the annotated subset of ImageNet, which comprises 38% of the entire dataset. Within this subset, we filter out images with overlapping instances where the IoU exceeds 10%, based on the bounding box annotations. This filtering process results in 6% of the annotated subset being used to train our proposed method. Consequently, our approach utilizes only 35% of the images employed by the baseline. 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 approach and the baseline using the COCO 2017 Evaluation 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 method follows 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 to minimize resource requirements. The key distinction between the baseline and our approach lies in the dataset size: our method utilizes only 35% of the images and their corresponding MaskCut annotations compared to the baseline. The baseline is

trained over 160K iterations, while our method requires 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 bounding box and segmentation evaluation on COCO Eval datasets on models trained (without self-training) on all images and images without overlapping instances

COCO Eval	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 bounding box and segmentation evaluation on split COCO Eval dataset on 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 an observable improvement in the results. As it can be observed from Table 1, our method improve in both instance detection and instance segmentation tasks. The table shows AP and AP50 of both instance detection (bbox) and instance segmentation (segm) tasks on COCO Eval datasets 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

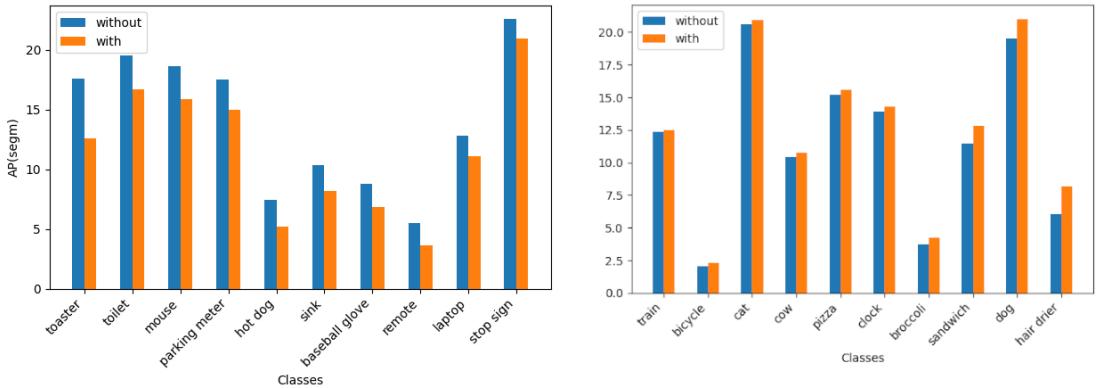


Figure 18: Class-wise comparison of change in AP_{segm} of models trained using images with and without overlapping instances evaluated on COCO Eval dataset - 10 most and least improved classes

in section 4.4.3) with batch size 8, which can be observed from Table 5. It is to be noted that the evaluation is performed using class-agnostic annotation generated from the original 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 extend 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 - a problem that is difficult to address without introducing explicit semantic information, which is typically provided through semi-supervised methods. 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 Eval dataset. This analysis allows for a more granular

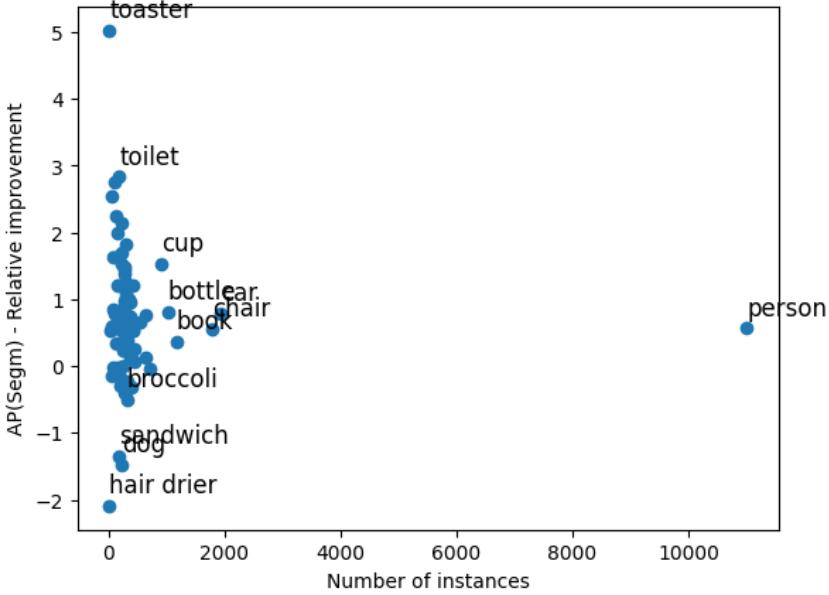


Figure 19: Relative Improvement (AP (Segm)) across Classes Against Number of Instances. Relative improvement of our model against the baseline on the y-axis and number of instances on the x-axis. Classes corresponding to extreme points are highlighted.

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

The test results are illustrated in Fig. 18, which highlights the 10 classes with the most significant improvements and the 10 classes with the least improvements in the COCO Eval dataset, 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. This distribution 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 3.1.2. ie, 1) Using all images of ImageNet 2) Only using object-centric images. We follow same training procedure as 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 lesser 5% lesser images compared to Experiment 4.4.1. It is due to removing the multi-instance images from the dataset. We use same baseline as Experiment 4.4.1 and models are evaluated on all images of COCO Eval datatset.

Results

As it can be observed from Table 3, our method improve in both object detection and instance segmentation tasks. The table shows AP and AP50 of both instance detection (bbox) and instance segmentation (segm) tasks on COCO Eval datasets evaluated using the baseline and our approach. Even by using just only 30% of the

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 Eval datasets on baseline and model trained with only object-centric images

training data, we could obtain comparable results with the baseline. However, the model trained without overlapping instances (Experiment 4.4.1) outperform this method by a small margin.

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. 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 rest 90% images constitute the training dataset for the second stage.

Training Procedure

The baseline follows training procedure as described in section 4.3. But our method comes with once extra step of refining MaskCut masks using our modified mask filtration method explained in section 3.2.2 and training from scratch again. This is followed 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).

Mask Filtration

In this section, we present a qualitative comparison of mask filtration methods, contrasting the results of the baseline approach with our proposed method.

Figure 20 illustrates the differences between the masks selected by our method and the baseline method across few example images. For most images with single instances with distinguishable background, there is no significant difference between masks selected by baseline and our method as given in Fig. 20 (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. 20 (b-d). In Fig. 20 (b-c) our approach successfully removes incorrect masks introduced by CutLER that are overlaid on better MaskCut masks. In Fig. 20 (d), despite both

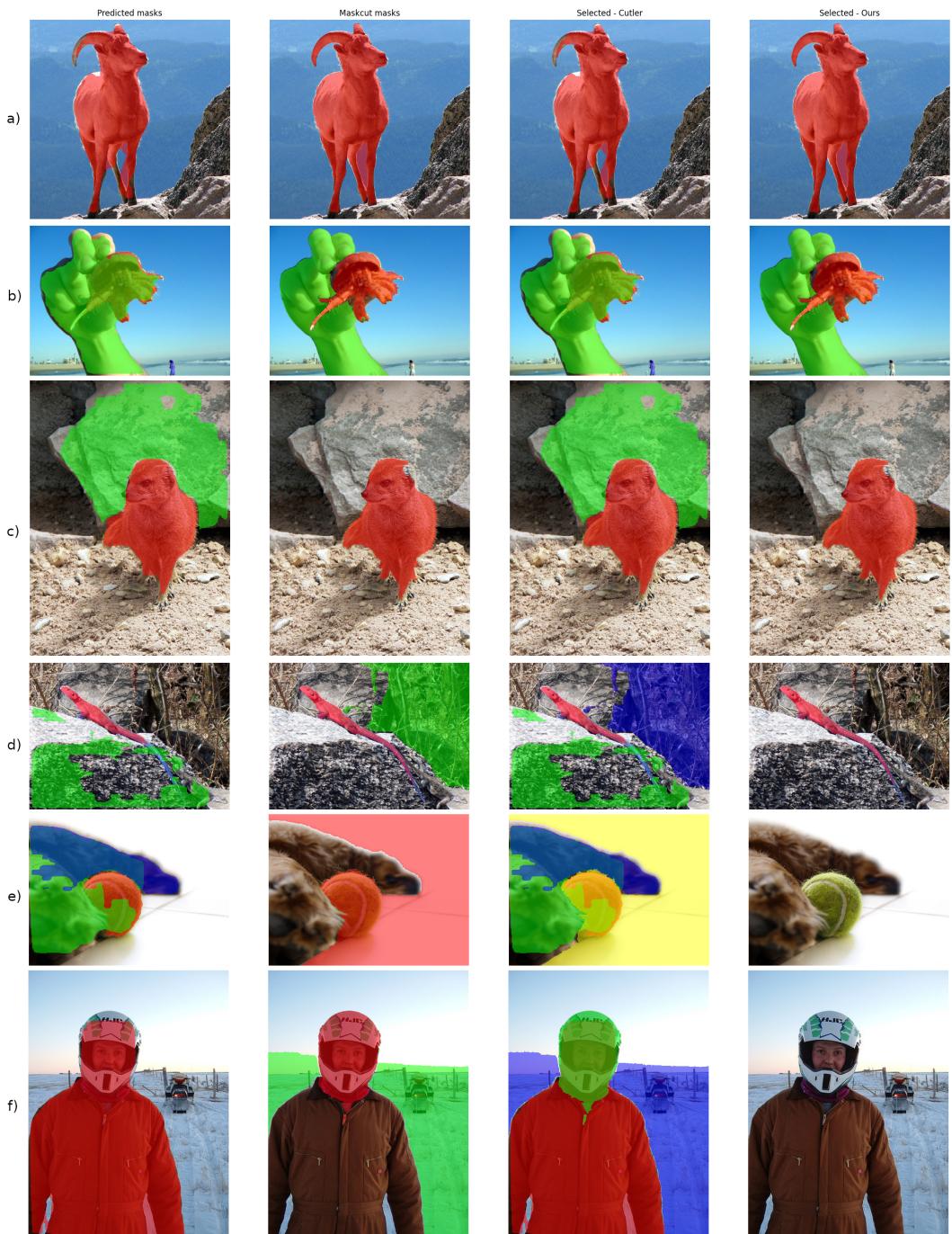


Figure 20: Comparison of filtered masks produced by the baseline and our method. Examples illustrate Cutler prediction masks, MaskCut masks, masks selected by Cutler and masks selected by our method respectively (left to right)

MaskCut and CutLER containing noisy masks, our method effectively filters out these imperfections.

As we discussed earlier in section 3.2.2, some images are dropped from the training dataset for not having any selected masks. In Fig. 20 (e), the image is dropped due to no selected masks, but CutLER prediction and mask selection are also bad. But in Fig. 20 (f) CutLER selection is certainly better compared to predicted masks. But the image will be dropped in our method. These examples reaffirms that we compromise on quantity for quality in our approach. But nevertheless our method outperforms baseline even with lesser data.

Using Our Mask Filtration Strategy for Self-Training

		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 4: AP_{box} and $AP50_{box}$ for Training and Self-Training evaluated on COCO Eval dataset (Batch size 8) using our mask filtration for self-training

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 4. This however was expected, as our masks are already less and more refined than baseline.

On the other hand, using masks generated using 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.

Combining Mask Filtration Strategies

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

Our evaluations span across five datasets, including 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 are summarized in Table 5 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.

		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 5: AP_{box} and $AP50_{box}$ for Training and Self-Training evaluated on COCO Eval dataset (Batch size 8) using baseline mask filtration for self-training

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 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 overfitting or the model’s diminishing returns from further self-training iterations, 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

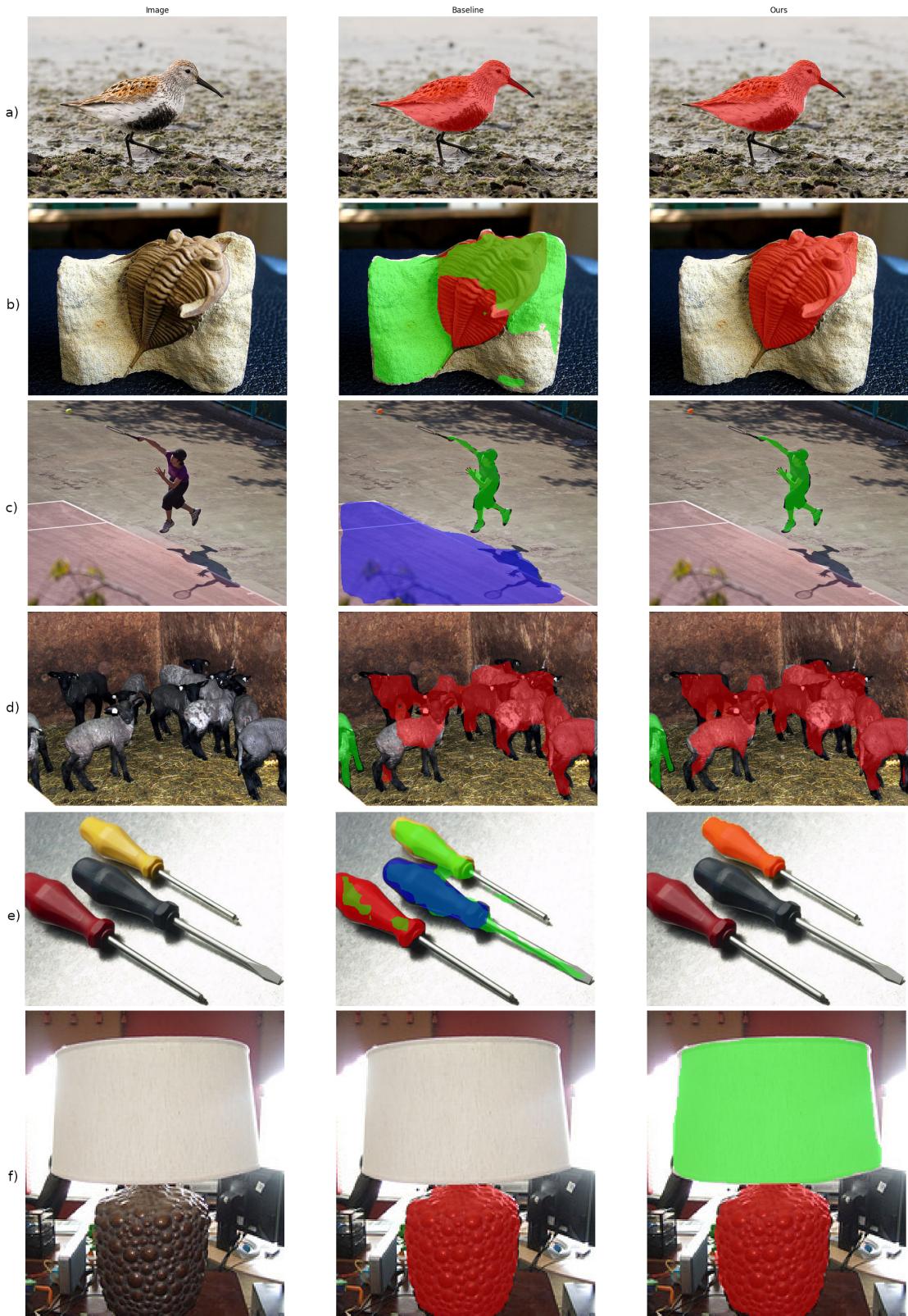


Figure 21: Qualitative results - Baseline vs Ours Masks predicted by CUTLER and Our model trained with modified mask filtration

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 21 presents a qualitative comparison between our method 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. 21 (a). As our method is trained with small number of filtered masks, our predictions mostly have lesser number of relatively better quality masks (Fig. 21 (b-c)). Grouping of the instances remains unsolved in both cases(Fig. 21 (d)). However, our method generates more complete masks with finer edges compared to the baseline. As we use only less masks for training due to our mask filtration method, our model often generates less number of masks compared to the baseline. In Fig. 21 (e), our model doesn't predict any masks, but baseline model predicts 4. However, there are few exceptions like Fig. 21 (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

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 6, 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

		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 6: AP_{box} and $AP50_{box}$ for Training and Self-Training evaluated on COCO Eval dataset for batch sizes 4 and 8

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 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 7: AR_{100} for the Best Models of Baseline and Our Method (Models from the First Self-Training Round) evaluated on COCO Eval dataset for segmentation (segm) and detection (bbox) tasks.

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.

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, which are often misclassified as objects in traditional mask generation methods.

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 contributes to the ongoing development of unsupervised learning methods by addressing critical limitations within the CutLER framework. While the improvements made in this work represent a significant step forward, they also highlight the need for continued research into more robust techniques for handling complex visual environments. Future work could explore the integration of more sophisticated mask refinement strategies and the extension of these methods to broader domains, ultimately pushing the boundaries of what can be achieved in unsupervised object detection and segmentation.

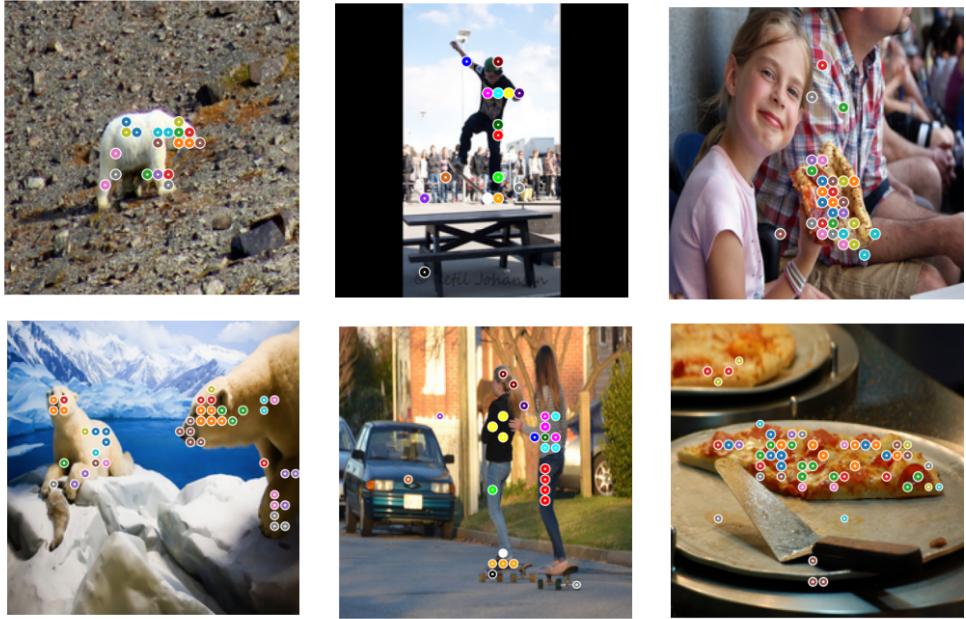


Figure 22: Keypoint correspondences using relaxed best buddies. Prototype images at the top and query images at the bottom. Same colored points represent similar features

5.1 Future Work

5.1.1 Keypoint Correspondences

For a good part of the thesis work, we worked with keypoint correspondences extracted from similarity matrix of key descriptors of two images to extract semantically similar parts in the image and use this information to separate the instances. We extracted the instances using keypoint correspondences taking inspiration from the SuperGlue [39] paper. But the main challenge was instead of working with images taken two different perspectives, we are dealing with entirely different pair of images with similar instances.

Figure 22 shows the keypoint correspondences between the prototype and query images. The correspondence is calculated by applying the relaxed best buddy algorithm on the descriptors corresponding to the foreground part of the prototype and query images, which is selected by applying a threshold on the saliency map. We relaxed the original best buddy algorithm [40] to extract more correspondence points enough to form a graph to perform graph-cut similar to [5, 39]. We explored our options to geometrically separate the instances using the semantic information

in the correspondences. But as we were unable to find a promising approach or inspiration from our research. For instance, Integer programming for multidimensional assignment problem [41] has a strict initial graph specification which restrict us to adapt it to our problem. By considering the limitations of implementing a semi-supervised method using keypoint correspondences, we focus mostly on improving the unsupervised instance detection and segmentation by diving into the current state-of-the-art method CutLER [3]; exploring its limitations and ways to improve them. Nevertheless we recognize the potential of leveraging keypoint correspondences for instance detection and believe this approach warrants further exploration in future research.

5.1.2 Sensitivity to Hyperparameters

The MaskCut and mask filtration processes within the CutLER framework are highly dependent on various hyperparameters, such as the threshold for maximum number of masks generated in MaskCut, IoU threshold for mask selection and the confidence score used to filter pseudo-ground truths. These hyperparameters play a crucial role in determining the quality of the masks and, consequently, the overall performance of the model in unsupervised instance detection and segmentation tasks.

Investigating the potential of adaptive or dynamic hyperparameter tuning, where the model adjusts these values during training based on performance metrics, could lead to more robust and generalized mask generation. Such future work would help in further refining the CutLER framework and potentially extend its applicability across a broader range of datasets and tasks.

5.1.3 Unsupervised Detection of Overlapping Instances

Based on our findings, a key area for future research will 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 Galessio, 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 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] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” 2023.
- [7] L. Weng, “The transformer family,” *lilianweng.github.io*, Apr 2020.
- [8] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” 2020.
- [9] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum contrast for unsupervised visual representation learning,” 2020.
- [10] Y. M. Asano, C. Rupprecht, and A. Vedaldi, “Self-labelling via simultaneous clustering and representation learning,” 2020.

- [11] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, “Unsupervised learning of visual features by contrasting cluster assignments,” 2021.
- [12] 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.
- [13] P. Engstler, L. Melas-Kyriazi, C. Rupprecht, and I. Laina, “Understanding self-supervised features for learning unsupervised instance segmentation,” 2023.
- [14] 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.
- [15] W. V. Gansbeke, S. Vandenhende, and L. V. Gool, “Discovering object masks with transformers for unsupervised semantic segmentation,” 2022.
- [16] 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.
- [17] 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.
- [18] 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.
- [19] 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.
- [20] 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.
- [21] W. V. Gansbeke, S. Vandenhende, and L. V. Gool, “Discovering object masks with transformers for unsupervised semantic segmentation,” 2022.

- [22] 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.
- [23] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, “Emerging properties in self-supervised vision transformers,” 2021.
- [24] S. Maji, N. K. Vishnoi, and J. Malik, “Biased normalized cuts,” in *CVPR 2011*, pp. 2057–2064, 2011.
- [25] 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.
- [26] Z. Cai and N. Vasconcelos, “Cascade r-cnn: High quality object detection and instance segmentation,” 2019.
- [27] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask r-cnn,” 2018.
- [28] 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.
- [29] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” 2015.
- [30] 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.
- [31] 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.
- [32] 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.

- [33] 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.
- [34] A. Ziegler and Y. M. Asano, “Self-supervised learning of object parts for semantic segmentation,” 2022.
- [35] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, “Self-training with noisy student improves imagenet classification,” 2020.
- [36] 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.
- [37] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The kitti dataset,” *International Journal of Robotics Research (IJRR)*, 2013.
- [38] 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.
- [39] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, “Superglue: Learning feature matching with graph neural networks,” 2020.
- [40] 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.
- [41] 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.

