

# DINO Features for Salient Object Detection

by

Albert Ding

A thesis  
presented to the University of Waterloo  
in fulfillment of the  
thesis requirement for the degree of  
Master of Mathematics  
in  
Computer Science

Waterloo, Ontario, Canada, 2024

© Albert Ding 2024

### **Author's Declaration**

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

## Abstract

In the field of computer vision, it can be useful to solve the problem of modeling the visually “salient” areas in a scene viewed by a human. One way to formulate this problem is to give an estimate of how likely a pixel is to belong to an “object” (as opposed to the background) which altogether forms a “saliency” mask. This has become a task commonly known as Salient Object Detection (SOD).

Supervised methods for this task are given a set of images and their corresponding pixel-precise saliency mask targets (often hand-labeled) and aim to learn the relationship between them. Unsupervised SOD methods in comparison attempt to identify salient objects by solely examining an image. This gives unsupervised methods the advantage of not needing expensive labels which are susceptible to human error. One good place to start an unsupervised method from is a technique called unsupervised feature learning. Interestingly, the Vision Transformer (ViT) [30] which struggled to find its place in the task of image classification, was able to produce features that are directly useful for SOD when it is trained in an unsupervised manner [11] called self-**DI**stillation with **NO** labels (DINO).

The authors of LOST [97] which investigates DINO features further have found that the “keys” from the last attention layer stand out as the most useful. Melas-Kyriazi *et al.* [80] and TokenCut [120] explore how applying spectral clustering methods to these features can be a good way to do tasks in computer vision without supervision. In this thesis we choose to continue to investigate these features for use in SOD by developing a K-means clustering-based method. From our experiments with a multitude of methods for using DINO features we observe that the clusters are rich in salient information. This could mean that the features themselves are particularly useful for SOD when processed by clustering methods.

We first apply K-means clustering to DINO features, then select some clusters to be salient according to various heuristics, upscale and post-process the resulting coarse saliency maps, obtaining pseudo-ground truth. Finally, we train a SelfReformer [136] saliency model on our pseudo-ground truth. The most important step of our approach is the heuristic which decides on salient clusters after K-means clustering. We have done an extensive development and evaluation of many heuristics, and, surprisingly, the simplest heuristic which assigns the clusters not touching the image border to be salient works the best, outperforming the complex method based on eigenvectors presented in [80], after SelfReformer training.

## **Acknowledgements**

First and foremost, I would like to thank my supervisor, Olga Veksler. I would also like to give thanks to Yuri Boykov and Jeff Orchard for agreeing to be the readers of my thesis. Finally, I want to thank Lori Paniak for helping me with WATGPU resources.

## **Dedication**

This thesis is dedicated to everyone.

# Table of Contents

|                                             |             |
|---------------------------------------------|-------------|
| <b>Author's Declaration</b>                 | <b>ii</b>   |
| <b>Abstract</b>                             | <b>iii</b>  |
| <b>Acknowledgements</b>                     | <b>iv</b>   |
| <b>Dedication</b>                           | <b>v</b>    |
| <b>List of Figures</b>                      | <b>viii</b> |
| <b>List of Tables</b>                       | <b>x</b>    |
| <b>1 Introduction</b>                       | <b>1</b>    |
| 1.1 Salient Object Detection . . . . .      | 1           |
| 1.2 Solving SOD . . . . .                   | 4           |
| 1.3 Unsupervised Feature Learning . . . . . | 5           |
| 1.4 Contribution . . . . .                  | 7           |
| 1.5 Thesis Structure . . . . .              | 8           |
| <b>2 Background</b>                         | <b>9</b>    |
| 2.1 Neural Networks . . . . .               | 9           |
| 2.2 Unsupervised Feature Learning . . . . . | 14          |

|          |                                                 |           |
|----------|-------------------------------------------------|-----------|
| 2.2.1    | DINO                                            | 15        |
| 2.2.2    | DINO Features                                   | 16        |
| 2.3      | Salient Object Detection (SOD)                  | 19        |
| 2.4      | Spectral Methods                                | 20        |
| <b>3</b> | <b>Methods</b>                                  | <b>22</b> |
| 3.1      | DINO Feature Extraction                         | 22        |
| 3.2      | Local K-Means Clustering                        | 23        |
| 3.3      | Global K-Means Clustering                       | 26        |
| 3.4      | Unreliable Cluster Filtering                    | 28        |
| 3.5      | Salient Cluster Determination: Local Heuristics | 30        |
| 3.5.1    | Center Prior                                    | 30        |
| 3.5.2    | Least Border                                    | 30        |
| 3.5.3    | Local Spatial Centeredness                      | 32        |
| 3.5.4    | Color Histogram                                 | 33        |
| 3.6      | Salient Object Determination: Global Heuristics | 34        |
| 3.7      | Post-processing                                 | 35        |
| 3.8      | Training a Supervised Network                   | 37        |
| <b>4</b> | <b>Experiments</b>                              | <b>40</b> |
| 4.1      | Benchmark Dataset                               | 40        |
| 4.2      | Evaluation Metrics                              | 41        |
| 4.3      | SelfReformer Training                           | 42        |
| 4.4      | Pseudo-Ground Truth Accuracy                    | 44        |
| 4.5      | SelfReformer Training Results                   | 47        |
| 4.6      | Ablations                                       | 49        |
| 4.7      | Qualitative Results                             | 51        |
| <b>5</b> | <b>Conclusion</b>                               | <b>53</b> |
|          | <b>References</b>                               | <b>54</b> |

# List of Figures

|     |                                                                                                                                                                                                                                                                                                                                                                                                                                      |    |
|-----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 1.1 | General Purpose Image Segmentation [34]                                                                                                                                                                                                                                                                                                                                                                                              | 1  |
| 1.2 | Semantic Segmentation Example [14]                                                                                                                                                                                                                                                                                                                                                                                                   | 2  |
| 1.3 | SOD Image and Ground Truth Examples [115]                                                                                                                                                                                                                                                                                                                                                                                            | 3  |
| 1.4 | Attention maps produced by DINO [11]                                                                                                                                                                                                                                                                                                                                                                                                 | 6  |
| 2.1 | Perceptron Visualization [89]. The heaviside step function can be replaced with any “activation” function which is meant to model the expected electrochemical activation behaviour a neuron might exhibit in an organic brain.                                                                                                                                                                                                      | 10 |
| 2.2 | MLP Visualization [4]                                                                                                                                                                                                                                                                                                                                                                                                                | 10 |
| 2.3 | CNN Visualization [106]                                                                                                                                                                                                                                                                                                                                                                                                              | 12 |
| 2.4 | Query Tokens and Key Tokens [1]                                                                                                                                                                                                                                                                                                                                                                                                      | 13 |
| 3.1 | K-Means Baseline [80] Initial Investigation Results<br>From left to right - Image, Ground Truth and $K = 2$ Baseline Prediction                                                                                                                                                                                                                                                                                                      | 24 |
| 3.2 | Local Clustering Example with $K = 5, 10$ and $15$ on the DUTS dataset [115]                                                                                                                                                                                                                                                                                                                                                         | 25 |
| 3.3 | Cluster Visualisation. From left to right we have the input image, the ground truth, the clusters produced by the local clustering step, and global clustering. In the fifth column, we show the results of global clustering after unreliable cluster filtering. Unreliable clusters are shown in white. In the first row we can see that the two clusters originally comprising the dog were merged by the global clustering step. | 27 |
| 3.4 | Illustration of our criterion for cluster similarity. The points in two clusters are shown with red and blue. Cluster centers are shown with black.                                                                                                                                                                                                                                                                                  | 28 |

|      |                                                                                                                                          |    |
|------|------------------------------------------------------------------------------------------------------------------------------------------|----|
| 3.5  | From left to right - Image, Ground Truth, Local Clustering, Center prior (border-based), Center Prior (corner-based) . . . . .           | 31 |
| 3.6  | From left to right - Image, Ground Truth, Local Clustering, Without Unreliable Cluster Filtering, Unreliable Cluster Filtering . . . . . | 31 |
| 3.7  | From left to right - Image, Ground Truth, Local Clustering, Least Border Heuristic . . . . .                                             | 32 |
| 3.8  | From left to right - Image, Ground Truth, No Crf, With CRF . . . . .                                                                     | 37 |
| 3.9  | From left to right - Image, Ground Truth, Pseudo-ground Truth . . . . .                                                                  | 38 |
| 3.10 | From left to right - The Original Image, Ground Truth, Object Border Background and 80% Random Void . . . . .                            | 39 |
| 3.11 | From left to right - Input Image, Ground Truth, Prediction without pre-training, Prediction with pretraining . . . . .                   | 39 |
| 4.1  | From left to right - Input Image, Ground Truth, Prediction by Main Method, Prediction by Local Spatial Centeredness . . . . .            | 52 |

# List of Tables

|     |                                                                                                                                                                                                                                                                                                                                                      |    |
|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 4.1 | The accuracy of pseudo-ground truth of our main methods compared to the eigenvectors [80] method, and eigenvectors [80] methods with spatially adjusted salient regions. Set size is the number of training images. Our method discards some of the training images which results in a smaller training set. . . . .                                 | 45 |
| 4.2 | Performance comparison of local $K = 2, 5$ and $10$ on the training set, DUTSTR. . . . .                                                                                                                                                                                                                                                             | 46 |
| 4.3 | Performance comparison of global $K = 5, 10, 20$ and $50$ on the training set, DUTSTR. . . . .                                                                                                                                                                                                                                                       | 46 |
| 4.4 | Main method variations performance comparison. LSC: Local spatial centeredness from Section 3.5.3. LSCU: Local spatial centeredness with unreliable cluster filtering from Sections 3.5.3 and 3.4 respectively. GSC: Spatial Centeredness Global Heuristic from Section 3.6. GBR1: Global Border Ratio 1 from Section 3.6. . . . .                   | 47 |
| 4.5 | Performance comparison of ablation ground truth. The 2% size filter did not change the ground truth when not using a CRF. . . . .                                                                                                                                                                                                                    | 47 |
| 4.6 | Post-training performance comparison for main method variations. LSC: Local spatial centeredness from Section 3.5.3. LSCU: Local spatial centeredness with unreliable cluster filtering from Sections 3.5.3 and 3.4 respectively. GSC: Spatial Centeredness Global Heuristic from Section 3.6. GBR1: Global Border Ratio 1 from Section 3.6. . . . . | 48 |
| 4.7 | Post-training performance comparison for main method and supervised baseline. Supervised: Training with the actual DUTSTR ground truths. . . . .                                                                                                                                                                                                     | 48 |
| 4.8 | Post-training performance comparison for main method and no pretraining baseline. No Pretraining: Training on the main method without a pretrained PVTv2 backbone. . . . .                                                                                                                                                                           | 48 |

|      |                                                                                                                                                                            |    |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 4.9  | Post-training performance comparison for main method and eigenvectors. .                                                                                                   | 49 |
| 4.10 | Loss function comparison. The original loss and BCE were used with void pixels and 20% random background labels while CCE was used with the indicated soft labels. . . . . | 49 |
| 4.11 | Void pixel ablation. . . . .                                                                                                                                               | 50 |
| 4.12 | Local K ablation. . . . .                                                                                                                                                  | 50 |
| 4.13 | Global K ablation. . . . .                                                                                                                                                 | 50 |
| 4.14 | Unreliable Cluster Filtering ablation. . . . .                                                                                                                             | 50 |
| 4.15 | CRF ablation. Note - The 2% size filter did not change the ground truth when not using a CRF. . . . .                                                                      | 51 |
| 4.16 | Corner-based center prior ablation. . . . .                                                                                                                                | 51 |
| 4.17 | DUTSTE Baseline. The performance of the main method, default eigenvectors and spatially adjusted eigenvectors. . . . .                                                     | 51 |

# Chapter 1

## Introduction

### 1.1 Salient Object Detection

Image segmentation is one of the main problems in computer vision today. In image segmentation, a digital image is partitioned into sets of pixels called segments. In classical computer vision (before deep learning), one often solved the task of general purpose image segmentation [34]. In general purpose image segmentation, the task is to partition an image into segments, where pixels in the same segment are expected to be similar to each other in some way, for example pixels can have similar colors, or intensities, or texture [99]. One way to mark segments is to assign an integer label to every pixel so pixels with the same label belong to the same segment. Image segments can also be marked by giving a set of contours whose enclosed regions are interpreted as the segments. See Figure 1.1 for an example of general purpose image segmentation.

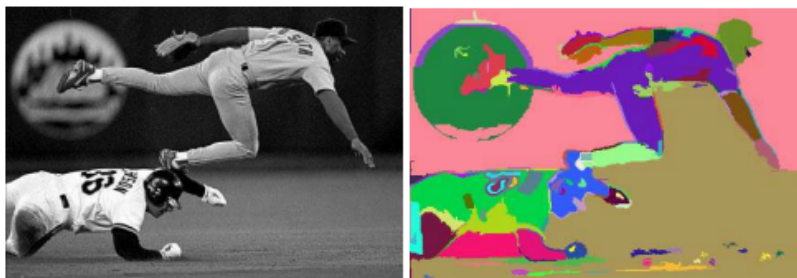


Figure 1.1: General Purpose Image Segmentation [34]

Most often, we want image segments to correspond to objects or well defined object parts. General purpose image segmentation does not guarantee that, since what constitutes a segment is defined in terms of the color/texture/etc. similarity of pixels inside a segment. An object can have pixels of distinct colors/textures, and conversely, there can be different objects of similar color/textures. Semantic segmentation [96, 14, 52], is a special type of image segmentation where each pixel is labeled with an object class (i.e. the semantic category). Thus in semantic image segmentation, the task is to partition an image into segments where each segment consists of pixels of the same object class it belongs to. Usually, there is a set of classes of interest which one wants to segment (i.e. people, cars, etc.), and all other classes are grouped together into a “background” class. See an example of semantic segmentation in Figure 1.2, where the set of object classes is {person, bicycle}. If there is more than one object from the same class in an image, all these objects are assigned the same class label. This is as opposed to another way of formulating image segmentation called instance segmentation where the goal is to have a separate label for each individual object belonging to the same class.



Figure 1.2: Semantic Segmentation Example [14]

There is a special useful case of semantic segmentation, where there are only two classes: the “object” class and the background class. This task is called salient object detection (SOD). This task results in producing a binary mask where each pixel is labeled as either the object (usually visualized as bright) or the background (usually visualized as dark), see Figure 1.3. Often, in addition to the mask, one produces a confidence score of how likely each pixel belongs to the salient object, or, in other words, the degree of visual saliency of a pixel. Historically, SOD research began with cognitive psychology research [104], from which arose a model of human visual attention which proposed that certain regions of a

scene would be first to attract attention from a human viewer, and thus are “salient” or prominent in the scene. However, there is no rigorous definition of visual saliency, only an intuitive one.

Before deep learning, SOD was often solved by designing by hand, properties which salient regions are likely to possess, such as central location [74], high contrast on the boundary [3], etc. However, these approaches are not robust and too simplistic to accurately solve the problem. More successful approaches are based on learning SOD from images where the salient objects are annotated by humans. In particular, the most successful methods are based on deep learning [116, 136].

SOD has many applications to tasks in computer vision such as image captioning [132, 33], semantic segmentation [124, 118] and object detection [139]. In particular, object detection is directly responsible for helping real-world vision tasks such as autonomous driving [125] and robotics [38]. SOD has even been used for AR [88] and can be a preprocessing step to many different tasks in computer vision.

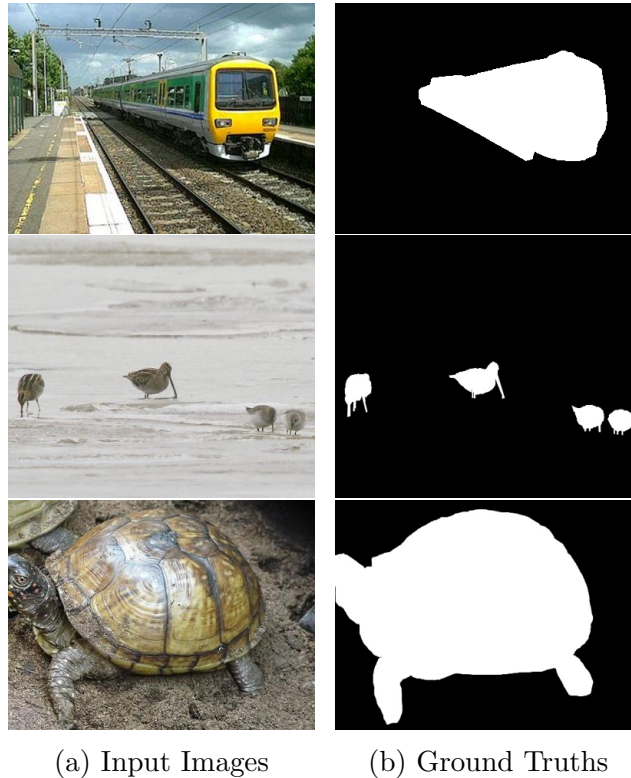


Figure 1.3: SOD Image and Ground Truth Examples [115]

## 1.2 Solving SOD

Early classical SOD methods are based on designing by hand features that have high response to salient objects. For example, in [74], they designed 3 features which they call multi-scale contrast, center-surround histogram and color-spatial distribution. Multi-scale contrast is computed by taking a linear combination of the contrast of an image at multiple scales. The idea behind the center-surround histogram, where they take the difference between the color histograms of a candidate “center” rectangle and an area of equal size found inside a “surrounding” rectangular boundary, is to make use of the fact that salient objects can be easily discerned from the area surrounding their boundaries. Finally, the color-spatial distribution feature is defined by representing all colors of an image with a GMM and computing the spatial variance of those colors so that the final feature value assigned to pixels belonging to a certain color is inversely proportional to the computed spatial variance. To further concentrate the saliency prediction done by color-spatial distribution, they make use of the center prior (which is the tendency for an object to appear at the center of an image), and make it biased towards the center.

In another approach, [21], they compute a color histogram after quantizing the color space (to reduce the number of possible colors) and used it in two contrast-based methods of SOD called Histogram-based contrast (HC) and Region-based contrast (RC). For HC, the main principle is to compute the saliency of a pixel as the sum of its color differences with all other pixels. Since the color histogram was computed in the compressed quantized space, the computational complexity was greatly reduced. The final post-processing step involves smoothing the color space by taking a weighted average of similar colors. For RC, the main idea is to take the color histograms of regions produced by an image segmentation algorithm and define the saliency of a region as a weighted sum of the color difference between all other regions. Although the color histograms are normalized to act as a probability distribution, the weights used are proportional to the number of pixels in a region.

While both contrast and center priors have proven to be useful priors, Wei *et al.* [123] proposed the use of priors that can be found in the background which they call the boundary prior and the connectivity prior. The boundary prior is based on the assumption that the boundary of an image is made up of mostly the background and the connectivity prior is based on the assumption that background segments can be connected easily because of their homogeneity. Armed with these two priors, they developed an algorithm which they call geodesic saliency.

Classical approaches to SOD are based on hand-crafted features and priors and cannot solve the problem to a high degree of accuracy. The area of SOD has been revolutionized

by deep learning [66], with one of the first fully supervised SOD deep learning approaches in [68]. In fully supervised SOD one needs a dataset where salient objects have been labeled by a human. Then, a CNN architecture is designed, where the CNN outputs, for each pixel, the probability of that pixel to belong to a salient object. Various CNN architectures tailored to the SOD problem have been designed [147, 63, 116].

Most recently, Transformer[108]-based models have started to gain traction in computer vision. Transformers often outperform CNNs due to the presence of attention layers, which can model longer-range interactions between pixels more successfully than CNNs. One Transformer-based model of particular interest is the Vision Transformer (ViT) [30], and one of the latest examples of using Transformers for SOD is the SelfReformer [136].

The ViT has recently been modified by Caron *et al.* [11] with a method of self-supervision called self-**D**Istillation with **N**O labels (DINO) and some of the features (also referred to as the attention maps) from the last layer exhibited saliency from a qualitative standpoint. Building on this foundation, Melas-Kyriazi *et al.* [80] have found these features to be quite powerful for unsupervised semantic segmentation. In this work, we will explore the potential of self-supervised features from DINO to generate saliency masks as well as the effectiveness of using those masks as pseudo-ground truths.

It is empirically evident that this revolution has been enabled by the amount of data that has pixel-level annotations. While supervised methods of training neural networks work well [136], providing adequate ground truths for training can be very expensive as each pixel of an image must be labeled. Some have tried to lower the amount of supervision needed with semi-supervised learning [76] and weakly-supervised learning [90]. Avoiding this challenge of labeling has become one of the main motivations behind researching unsupervised methods, which is the ultimate goal of this work.

### 1.3 Unsupervised Feature Learning

From the advent of deep learning also came feature learning. Feature learning usually involves extracting from certain hidden layers of a neural network and designating the activations of these layers as deep features. Feature learning is also called representation learning because the extracted features can be considered a representation of an image.

Most feature learning methods are supervised, i.e. features are learned while performing some task, such as classification, using labeled data (ground truth) [26]. However, unsupervised feature learning is an attractive alternative to feature learning, as there is overwhelmingly more unlabeled data that can be used for training and normally, the more

training data there is, the better the learning outcome. For unsupervised learning, one has to come up with some pretext task and a loss function for training. For example, the task might be to determine whether two input images are crops/augmented views of the same image or come from distinct images [46]. Recently, [11] proposed an unsupervised feature learning method based on the Transformer architecture which excels at feature learning. We use DINO features as the basis of our approach for SOD without pixel-precise supervision.

Normally, the input to a Transformer is a sequence of “tokens”, which are equal-length vectors encoding the input text of some human language. To adapt such an architecture for computer vision, the ViT breaks up an image into patches of 16x16 pixels to form a sequence of tokens. The output of the Transformer which is commonly referred to as an attention map gives an “activation” value, a scalar, for every possible pair of tokens. The hope is that these activations are a result of the Transformer’s ability to model spatially long-range dependencies between the patches of the input image.

In the case of DINO, a prepended classification (CLS) token (an idea borrowed from [27]) has its dot product taken with the rest of the non-classification tokens producing a sequence of activations that can be reinterpreted as a downscaled representation of the input image. Applying nearest-neighbour interpolation to upscale the attention map back to the input image’s original size appears to show that DINO understands image segmentation. The attention maps from DINO shown in Figure 1.4 are self-supervised features of particular interest for SOD as discovered by [97, 80, 120, 75]. Since DINO uses 6 attention “heads” (instances of a Transformer), the maps shown could come from any one of those 6 heads.

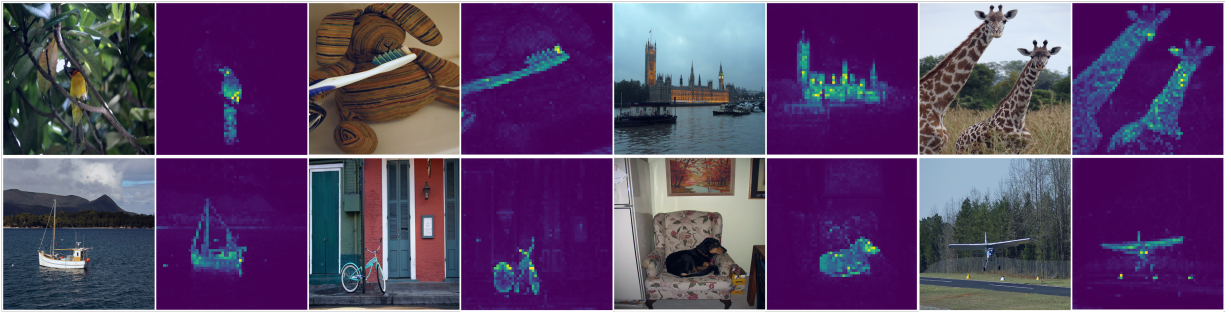


Figure 1.4: Attention maps produced by DINO [11]

## 1.4 Contribution

In this thesis, we explore unsupervised Salient Object Detection (SOD). The advantage of an unsupervised approach is that there is much more unsupervised data available for learning compared to labeled (supervised) data. Another advantage is that by using unsupervised data, we have the possibility of avoiding any biases the annotator might have when they perform labeling.

Our approach to unsupervised SOD is based on DINO features. DINO features have been shown to be useful for segmentation, but up to date, there is no comprehensive exploration for using DINO features for the task of SOD.

We first evaluate an approach based on eigenvectors [80], which was proposed in prior work for semantic segmentation with DINO features.

Then we develop and evaluate several simpler approaches. We start by performing K-means clustering on DINO features and show that the resulting clustering breaks images into meaningful segments, where one (or several) of the segments correspond to the salient object or salient object parts. The main problem then becomes determining which of the segments correspond to the salient object or a salient object part.

First we develop an approach for filtering out unreliable clusters. These are clusters which are not frequent, i.e. they do not have many members across the whole training dataset. After unreliable clusters are removed, we propose several simple approaches for determining salient objects (or parts of salient objects).

The first approach is based on simply assigning any segment that does not touch the image border to the salient object. This is based on the well known fact that salient objects tend to occupy the image center which we will call the center prior

Our second approach also utilizes the center prior and selects clusters that are close to the image center. The selection process can be done through a variety of methods such as top k closest clusters, or just taking the cluster assignment which has the largest difference in average distance between the object and background pixels.

Finally, our third approach is based on applying “global” K-means clustering to the set of all “local” cluster centers (the cluster centers produced from applying K-means clustering to each image individually) across the dataset. We reassign each local cluster to a global cluster based on which global cluster their cluster center was assigned to. The collection of local clusters belonging to a specific global cluster can then be assigned to be salient or not based on global cluster statistics (dataset-level information) such as applying a threshold to the count of the number of local clusters touching the border. The best results for

approaches based on global cluster statistics came from taking the global clusters with the lowest average border-touching pixels to cluster size (as opposed to border size) ratios.

The final step in all of our approaches is to train a Transformer-based neural network [136] on a dataset with “pseudo”-ground truths, i.e. a dataset where the ground truth labeling is provided by the estimated salient object. Such pseudo-ground truths are noisy, and we develop a method which is more successful for training from noisy ground truths.

Surprisingly, our first approach, where we simply label any segment not touching the image border as salient, leads to the best performance after using the results of this approach as pseudo-ground truth for neural network training.

## 1.5 Thesis Structure

An introduction to unsupervised SOD is given in Chapter 1. In Chapter, 2 we go over the history of unsupervised feature learning. Then, in Chapter 3, the main proposed methods for unsupervised SOD are discussed. Chapter 4 is dedicated to the experiments and results as well as some analysis of the results. Finally, a conclusion is given in Chapter 5.

# Chapter 2

## Background

### 2.1 Neural Networks

Neural networks began with the idea of modeling neural activity using propositional logic in 1943 [79]. This led to the birth of the perceptron in 1957 [91] with the goal of simulating the function of a biological neuron. The mathematical formulation of a perceptron is that of a function that takes as input a vector of numbers and outputs a linear combination of the input vector’s components. The coefficients are unanimously referred to as “weights” in the literature. These weights are said to be “learnable” because they are usually adjusted given an input dataset by gradient descent. Adjusting the weights of a neural network is also referred to as “training” the neural network. Traditionally, depending on the learning goal of a neural network, an appropriate loss function is chosen that measures the amount of loss the neural network has on a given input data point. The loss function is also the first point of differentiation when applying gradient descent. If training is done by computing the loss between the neural network’s output and a given expected output, then the neural network is known to have undergone training with “supervision”. See Figure 2.1 for a visualization of a single neuron (perceptron).

When the outputs of a set of perceptrons are used as the subsequent input to another perceptron, we call the resulting architecture and one of the earliest neural networks, a multi-layer perceptron (MLP). The first and last layers are known as the input and output layers respectively, and the rest of the layers are called hidden layers. See Figure 2.2 for a visualization of an MLP. Extending a neural network by adding more layers could help it model a wider set of functions compared to one with a single layer [81]. The more layers a neural network has, the “deeper” it is considered to be which is why

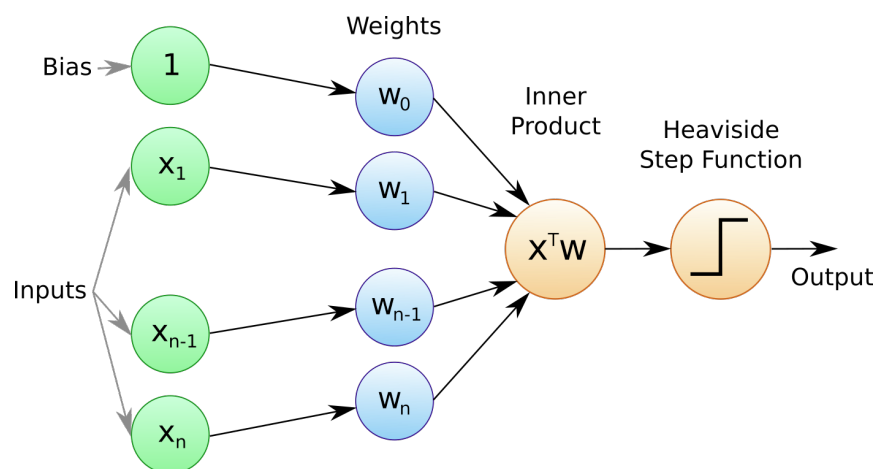


Figure 2.1: Perceptron Visualization [89]. The heaviside step function can be replaced with any “activation” function which is meant to model the expected electrochemical activation behaviour a neuron might exhibit in an organic brain.

the study of neural networks is generally known as “deep learning”. To help networks learn efficiently, the backpropagation algorithm which involves applying the chain rule and dynamic programming for gradient descent was essential [92].

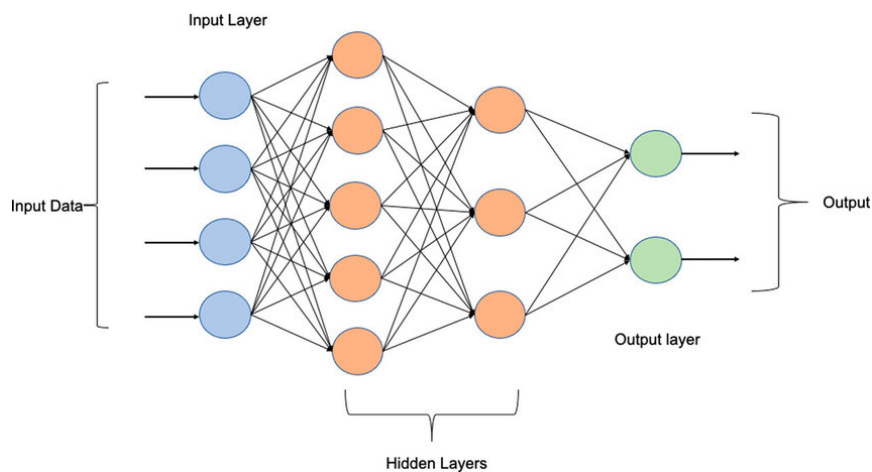


Figure 2.2: MLP Visualization [4]

## Convolutional Neural Networks (CNNs)

Convolutional neural networks [65] have a special structure that reduces the number of trainable parameters by taking advantage of the fact that features which are useful in one part of the image are often useful in other parts of the image, or so-called translational invariance. CNNs introduce a layer known as the convolutional layer, an alternative to using perceptrons which are also known as fully connected layers since there is a separate learnable weight parameter for every possible input and output neuron combination. A fully connected layer’s parameters can essentially be encoded as a learnable matrix where the elements of the matrix are “learned” iteratively through gradient descent. A convolutional layer on the other hand can give a massive reduction in the total number of parameters by “sharing weights”. Given an image, the weights are parameterized by a “kernel” which convolves over the image like a sliding window. A conventional convolution kernel has the same number of channels as the input image. After taking the dot product with every patch of the image the kernel is designated, a convolutional layer produces an output consisting of a single channel. To increase the size of the output of a convolutional layer in the channel dimension multiple kernels are used so their outputs can be concatenated along the channel dimension. See Figure 2.3 for an example of a convolutional layer. Convolutional layers can also be said to model the scope of image processing tasks better than a regular perceptron as the sequence of kernels of consecutive convolutional layers can be interpreted as having the ability to consolidate information over an increasingly larger receptive field. Early CNNs still used fully connected layers at the end to make a prediction but later fully connected layers were interpreted as a special case of a convolutional layer which allowed for Fully Convolutional Networks (FCNs) (a neural network that uses only convolutional layers) to come about. In 2011, a special type of CNN called AlexNet[61] was developed and it won the ImageNet 2012 [26] challenge by a large margin, bringing deep learning to the attention of the machine learning community.

## Transformers

One groundbreaking development in the field of deep learning was the creation of the Transformer [108] in 2016. Introduced to the field of AI for use in Natural Language Processing (NLP) with networks like BERT [27] and ALBERT [64], research efforts to bring Transformers into the domain of Computer Vision began most notably in 2020 with the Vision Transformer (ViT) [30] for image classification. Previously in the Introduction chapter, we discussed that the typical input to a Transformer designed to solve tasks in NLP is a sequence of equal-length vectors called tokens. The main reason behind

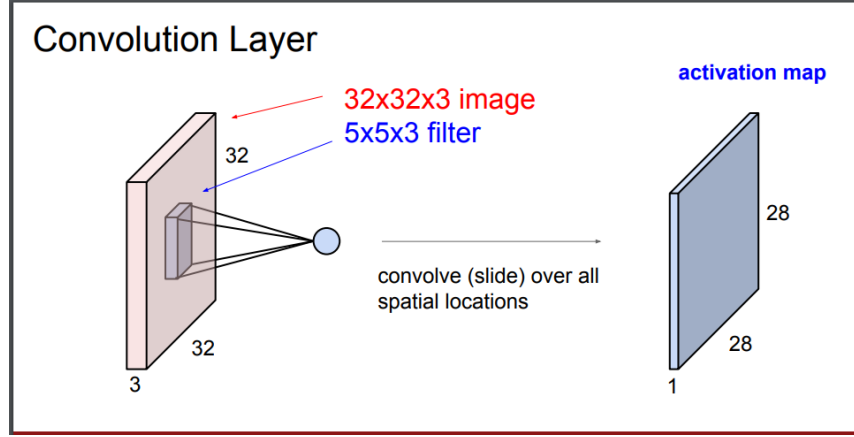


Figure 2.3: CNN Visualization [106]

the effectiveness of a Transformer is the attention mechanism which allows it to model long-range interactions more effectively compared to a CNN. The attention mechanism is implemented as follows. Recall that a transformer receives a sequence of tokens, which are then modified to another sequence of tokens at each internal stage of the transformer architecture. Each token of the sequence is put through learnable “query” and “key” matrices. The purpose of the query matrix is to modify the tokens in a way that allows the tokens to “query” the “key” tokens which the key matrix has hopefully modified so that each token can answer the query of a “query” token appropriately. This is achieved by taking the dot product of every possible query-key pair. See Figure 2.4 for an illustration.

A series of ViT-based models followed the ViT including Tokens-To-Token Vision Transformer (T2T-ViT) [135], DeepViT [149], CrossViT [13], PiT [49], LeViT [42], and CvT [126] for image classification, Segmentation Transformer (SETR) [148], Segmenter [98], and SegFormer [129] for semantic segmentation, Pyramid Vision Transformer (PVT) [117], Detection Transformer (DETR) [9], Unsupervised Pre-training DETR (UP-DETR) [25], and Deformable DETR [152] for object detection and Swin Transformer [77], and Twins [23] for all three. For SOD, Visual Saliency Transformer (VST) [73] and Depth-supervised Fusion Transformer (DFTR) [150] were created. Some examples of utilizing Transformers to solve SOD include SelfReformer [136], [80], TokenCut [120] and [75].

|                                                                           | $\vec{a}$                                                        | $\vec{\text{fluffy}}$                                            | $\vec{\text{blue}}$                                              | $\vec{\text{creature}}$                                          | $\vec{\text{roamed}}$                                            | $\vec{\text{the}}$                                               | $\vec{\text{verdant}}$                                           | $\vec{\text{forest}}$                                            |  |
|---------------------------------------------------------------------------|------------------------------------------------------------------|------------------------------------------------------------------|------------------------------------------------------------------|------------------------------------------------------------------|------------------------------------------------------------------|------------------------------------------------------------------|------------------------------------------------------------------|------------------------------------------------------------------|--|
|                                                                           | $\downarrow$<br>$\vec{E}_1$<br>$\downarrow_{W_Q}$<br>$\vec{Q}_1$ | $\downarrow$<br>$\vec{E}_2$<br>$\downarrow_{W_Q}$<br>$\vec{Q}_2$ | $\downarrow$<br>$\vec{E}_3$<br>$\downarrow_{W_Q}$<br>$\vec{Q}_3$ | $\downarrow$<br>$\vec{E}_4$<br>$\downarrow_{W_Q}$<br>$\vec{Q}_4$ | $\downarrow$<br>$\vec{E}_5$<br>$\downarrow_{W_Q}$<br>$\vec{Q}_5$ | $\downarrow$<br>$\vec{E}_6$<br>$\downarrow_{W_Q}$<br>$\vec{Q}_6$ | $\downarrow$<br>$\vec{E}_7$<br>$\downarrow_{W_Q}$<br>$\vec{Q}_7$ | $\downarrow$<br>$\vec{E}_8$<br>$\downarrow_{W_Q}$<br>$\vec{Q}_8$ |  |
| $\vec{a} \rightarrow \vec{E}_1 \xrightarrow{W_K} \vec{K}_1$               | +0.7                                                             | -83.7                                                            | -24.7                                                            | -27.8                                                            | -5.2                                                             | -89.3                                                            | -45.2                                                            | -36.1                                                            |  |
| $\vec{\text{fluffy}} \rightarrow \vec{E}_2 \xrightarrow{W_K} \vec{K}_2$   | -73.4                                                            | +2.9                                                             | -5.4                                                             | +93.0                                                            | -48.2                                                            | -87.3                                                            | -49.7                                                            | +7.8                                                             |  |
| $\vec{\text{blue}} \rightarrow \vec{E}_3 \xrightarrow{W_K} \vec{K}_3$     | -53.4                                                            | -5.7                                                             | +1.8                                                             | +93.4                                                            | -55.6                                                            | -56.0                                                            | -26.1                                                            | -62.1                                                            |  |
| $\vec{\text{creature}} \rightarrow \vec{E}_4 \xrightarrow{W_K} \vec{K}_4$ | -21.5                                                            | -29.7                                                            | -56.1                                                            | +4.9                                                             | -32.4                                                            | -92.3                                                            | -9.5                                                             | -28.1                                                            |  |
| $\vec{\text{roamed}} \rightarrow \vec{E}_5 \xrightarrow{W_K} \vec{K}_5$   | -20.1                                                            | -40.9                                                            | -87.8                                                            | -55.4                                                            | +0.6                                                             | -64.7                                                            | -96.7                                                            | -18.9                                                            |  |
| $\vec{\text{the}} \rightarrow \vec{E}_6 \xrightarrow{W_K} \vec{K}_6$      | -87.9                                                            | -33.3                                                            | -22.6                                                            | -31.4                                                            | +5.5                                                             | +0.6                                                             | -4.6                                                             | -96.8                                                            |  |
| $\vec{\text{verdant}} \rightarrow \vec{E}_7 \xrightarrow{W_K} \vec{K}_7$  | -41.2                                                            | -55.5                                                            | -42.3                                                            | -59.8                                                            | -79.0                                                            | -97.9                                                            | +3.7                                                             | +93.8                                                            |  |
| $\vec{\text{forest}} \rightarrow \vec{E}_8 \xrightarrow{W_K} \vec{K}_8$   | -58.9                                                            | -75.5                                                            | -91.1                                                            | -90.6                                                            | -75.6                                                            | -89.0                                                            | -70.8                                                            | +4.7                                                             |  |

Figure 2.4: Query Tokens and Key Tokens [1]

## SelfReformer

Yun *et al.* [136] begin by explaining that traditional deep SOD models employ an encoder-decoder setup, multi-scale features and global context provided by the decoder. They note that interpolation damages critical structural information that would have been useful for prediction and that the encoder also contains good global context information and aim to rectify these architectural decisions with 3 improvements to create SelfReformer using the PVTv2 [117] as their backbone. The goal of the improvements is to preserve details for even better performance in SOD. To address the issue of global context they add a Global Context Branch (GCB) on top of the encoder which has the job of preserving encoder features. They also create a Context Refinement Module (CRM) which helps multi-scale features to combine better and utilize Pixel Shuffle (PS) which preserves information during downsampling by shuffling the pixels into the channel dimension. Making these three improvements helped them achieve state-of-the-art performance on a multitude of

standard benchmarks for SOD.

## ViT Analysis

Some researchers have tried optimizing the training and architecture of the ViT such as Data-efficient image Transformers (DeiT) [102], [138], Class-Attention in Image Transformers (CaiT) [103], and T2T-ViT [135]. Tuli *et al.* [105] compared Transformers with CNNs and found that they are less biased towards textures. Naseer *et al.* [82] also studied properties of Transformers and came up with a similar observation. These observations combined with the work done by Caron *et al.* [11] on ViT to apply self-supervised learning, which they call self-DIstillation with NO labels (DINO), showing that the attention map features contain object segmenting information motivated [97, 80, 120, 75] to research new ways of using DINO features for various computer vision tasks.

## 2.2 Unsupervised Feature Learning

Learning features (representations) is considered by many to be one of the main reasons deep learning works so well [95]. Before deep learning, features in computer vision such as edges or texture were extracted from images by a group of methods known as feature detectors. When features are learned without supervision they can help alleviate the problem with supervised learning methods in computer vision of potentially requiring expensive pixel-level annotations [31]. One unsupervised method of learning is known as self-supervision. To tackle the problem of learning features with self-supervision directly, a number of pretext tasks have been proposed such as spatial context [28], jigsaw puzzles [85], rotations [41], inpainting [86], colorization [142], channel prediction [143] and contrastive predictive coding [48], [107]. The ultimate purpose of using a pretext task as a self-supervised learning goal is for a deep learning model to learn to produce features which are good image representations (that can then be used to solve the given pretext task) in the process of solving one.

Later research on self-supervised feature learning became more focused on contrastive learning [44], [127], SimCLR [17], MoCov2 [18], [58], [101], which is one approach that has started to see more success over pretext tasks. The foundation of contrastive learning as a form of self-supervision is the formulation of the loss function used. While a supervised method expects a fixed target to compute the loss, contrastive learning allows for a pair of dynamic targets that are usually augmented views of the input images. Many contrastive

learning methods involve both positive examples where both views come from the same image and negative examples where the network must determine that the views came from different images to prevent collapsing to a trivial solution. Some researchers have tried removing the need for negative examples SimSiam [19], BYOL [43], [137]. Since most of the previously mentioned methods of contrastive learning produce a single feature vector for image classification some researchers have explored dense contrastive learning [87], [119], [131] to learn more dense features for tasks like SOD and semantic segmentation. However, finetuning (further training) is still needed to adapt these dense contrastive learning methods to those tasks which means that the features they learned are not general enough for direct prediction.

Self-supervised ViTs such as SiT [6], DINO [11], MoCov3 [20], MST [71], BEIT [7], MAE [45] and SSL SWIN [130] have shown to be the most interesting way of applying self-supervision. Their performance on standard computer vision tasks such as object detection [39], MoCo [46], SwAV[10], BYOL [43], [40] is even better than that of CNNs, their most common supervised counterpart. One more possible use of unsupervised features is to generate the ground truths that will then be used in supervised learning has been explored by [84], [115], [141]. In particular [97, 80, 120, 75] have all tried to leverage self-supervised features extracted from the self-supervised ViT DINO for various tasks in computer vision.

### 2.2.1 DINO

In 2021, Caron *et al.* [11] came up with a self-supervised learning procedure which they call self-Distillation with NO labels (DINO). In chemistry, distillation is a process that separates a liquid mixture by boiling (raising the temperature) the liquid. Distillation in machine learning is a technique for compressing the knowledge of a large machine learning model or an ensemble of models into a single smaller model. Hinton *et al.* [50] do this by raising the “temperature” used in the softmax output layer of the large model to get a soft probability distribution so that its knowledge can be “distilled” into the smaller model.

Self-distillation is done in DINO by training a student model and a teacher model by giving them two random augmentations of the same image and asking the student to match the teacher. The teacher’s parameters are updated with an exponential moving average of the student’s parameters which make the teacher essentially a momentum encoder described by MoCo [46]. Unlike MoCo which uses a contrastive loss function to avoid collapse to a degenerate solution, DINO uses a standard cross-entropy loss and does “centering” and “sharpening” on the teacher’s output. Centering is essentially adjusting the teacher’s output by an exponential moving average of previous outputs while sharpening means using

a low temperature for softmax. The distillation parallel breaks down here since the teacher is supposed to be analagous to the larger model whose temperature is high in accordance to chemical distillation as well as the original distillation work done by Hinton *et al.*

By applying DINO to a ViT they obtain some interesting self-supervised learned features. They observed that these features qualitatively possess more segmenting information compared to ones produced by ViTs that have been trained with supervision. The features are also readily used in a K-nearest neighbour classifier for image classification, although the accuracy is slightly worse than just using a linear classifier (an MLP layer). They also perform several experiments which show the importance of using momentum contrast [46], multi-crop training SwAV[10], and small image patches (this makes sense intuitively as more information can be encoded by the attention activations at the end of each Transformer block). The attention maps shown in their paper are created specifically from upscaling the set of activations corresponding to the “query” classification token and every “key” non-classification token. To explore these high potential segmenting properties even further, several researchers specifically try to exploit DINO features for various computer vision tasks.

### 2.2.2 DINO Features

#### LOST

Firstly in 2021, Valeo.ai [97] used DINO features for object localization. The task of assigning a bounding box to the objects in an image is known as object discovery. The task becomes known as object detection when the bounding box is combined with a class label prediction. Specifically, Valeo.ai proposed a method for “Localizing Objects with Self-Supervised Transformers” abbreviated as LOST which exploits the activation features produced by DINO. They showed that LOST outperforms the state of the art in the CorLoc metric on the PASCAL VOC 2012 dataset and using the results as pseudo-ground truths for training a object detector lead to an even better performance. One of the crucial decisions they made was to use the “keys” features because the “keys” performed better compared to using the “queries” or the “values”. Since this decision was made based on empirical evidence, there is still plenty of opportunity to research exactly what makes the “keys” so much more powerful.

## Deep Spectral Methods

In 2022, Melas-Kyriazi *et al.* [80] showed applying spectral clustering to DINO features gives strong results in semantic segmentation and consequently SOD. Spectral clustering is a clustering method which is typically done by first creating a feature affinity matrix and then combining the eigenvectors corresponding to the smallest eigenvalues of the affinity matrix using some sort of post-processing. More specifically, they construct their affinity matrix from an RGB Image  $I \in \mathbb{R}^{3 \times H \times W}$  as follows:

1. They aggregate the color and spatial information to get a “color affinity” matrix  $W_{knn}$  with dimensions  $(HW) \times (HW)$  which encodes a fully-connected graph that treats each pixel as a node and each matrix entry denotes the edge weight between those pixels. The edge weight of edge  $u \rightarrow v$  is either  $1 - \|\psi(u) - \psi(v)\|$  if  $u$  is one of the  $k$ -nearest neighbours of  $v$  and 0 otherwise, where  $\psi(u)$  is a 5-vector consisting of the  $r, g, b$  intensities and  $x$  and  $y$  coordinates of pixel  $u$ .

2. To create the feature matrix  $W_{feat}$ , they use DINO “key” features which represent  $16 \times 16$  image patches and so have dimensions  $h \times w \times c$ , where  $h = \lfloor \frac{H}{16} \rfloor$ ,  $w = \lfloor \frac{W}{16} \rfloor$ , and  $c = 384$ . They treat the features,  $f$ , as a matrix with dimensions  $(h \times w) \times c$ . Then, they take the dot product of the features with the transpose to get a matrix that has dimensions  $(h \times w) \times (h \times w)$ . Finally, they threshold the resulting matrix at 0 to get rid of anti-correlations. This is done with  $\odot$  which is the standard notation for elementwise multiplication.

$$(W_{threshold})_i = \begin{cases} 1, & \text{if } (ff^T)_i > 0 \\ 0, & \text{otherwise} \end{cases}, \quad W_{feat} = ff^T \odot W_{threshold} \quad (2.1)$$

3. After bilinearly downsampling the color affinity matrix to  $(h \times w) \times (h \times w)$ , they add the two matrices to get,

$$W_{final} = W_{feat} + \lambda_{knn} W_{knn}, \quad (2.2)$$

where  $\lambda_{knn}$  is a hyperparameter to set the relative importance of the matrices.

To do SOD, they deem the simplest thing would be to take a threshold that assigns all positive values on the algebraic connectivity (the second eigenvector when ordering the eigenvectors by increasing eigenvalues) of the Laplacian of this affinity matrix to the salient object. Since each eigenvector has dimensions  $hw$ , they upsample it back up to  $HW$  with nearest neighbour interpolation and apply a CRF to get a saliency mask for each pixel of the input image at full resolution.

To do semantic segmentation, similarly to the method of object cosegmentation they apply K-means clustering to the eigenvectors to obtain segments for which they recomputed DINO features to be used again with K-means across the entire dataset for semantic labels. They also showed that their spectral clustering methods outperformed the state of the art. Again, the basis of the powerful results shown from this research is both the qualitative and empirical evidence that DINO features contain a significant amount of segmentation information.

## TokenCut

Similarly in 2022, Wang *et al.* [120] developed a method of utilizing DINO features to do object discovery, SOD and object detection. They also used a form of spectral clustering where the problem is first formulated as one that needs a graph cut and the graph is constructed is based on similarity scores of the features. The main difference is that Melas-Kyriazi *et al.* [80] essentially construct a fully connected graph whose edge weights are either the dot product of the features representing the endpoints of the edge or 0 if the dot product is negative, while TokenCut uses edge weights that are 1 if the cosine similarity of the pair of features is above a certain threshold  $\tau$  and *epsilon* (which they set to 1e-5) otherwise. Similarly to Valeo.ai, Wang *et al.* improved the performance by using the results to train a object detector. They showed that their method was able to improve upon Valeo.ai’s LOST method extended to unsupervised SOD, where they improved on the IoU metric compared to the state of the art on a multitude of datasets.

## Differentiable Normalized Cuts

Finally in 2024, Liu *et al.* [75] proposed a neural network based on DINO to do unsupervised dense prediction which includes tasks like semantic segmentation. Since the previously discussed methods all extract features from a frozen DINO backbone, they proposed using Differentiable Normalized Cuts to continue training. Their method consists of using a gradient formulation of Normalized Cuts (so that the gradients can be used in a self-supervised architecture based on DINO), and RoIAlign from Mask R-CNN [47] on two random crops of an image to do contrastive learning. They also have experimental evidence that this method can be applied to other pre-training methods and obtain state of the art performance in unsupervised dense prediction. The research done so far on the empirically and qualitatively evident salient information contained in DINO features has shown that these features are quite useful for solving a variety of tasks in computer vision.

## 2.3 Salient Object Detection (SOD)

### Supervised Deep Learning for SOD

Some of the earliest known direct research on SOD came from [74] and [3]. Non-deep SOD models such as [134, 94, 55, 59, 70, 78, 153, 133, 151] typically leveraged information contained in low-level features and heuristics. Deep SOD models later arrived to completely takeover and improve performance on the task. Early examples of deep learning applied to SOD include [114, 147, 16] and some later examples include [72, 67, 63, 116]. While early deep SOD models such as [68] still used fully connected layers, networks completely comprised of convolutional layers which are known as fully convolutional networks (FCNs) [69, 116] in the literature proved to be more performant. One of best performing attempts at SOD has been done by the Transformer-based SelfReformer [136].

### Unsupervised SOD

In this work, we are more interested in unsupervised methods of deep learning. Despite the great successes deep learning brought to performance in SOD, there were still concerns regarding the viability of training with full pixel-level supervision. First, let’s talk about unsupervised object localization. Previously, we discussed the tasks of object discovery and object detection which both fall under the classification of being object localization tasks. SOD can be considered a third object localization task. One way to do unsupervised object localization is to group images across image collections [51], [56], [57], [109] and has become a method known as object cosegmentation. This does not have good scalability, so extracting deep features from CNNs that have been pre-trained (with supervision) for the task of image classification has been explored by [24] and [144] for the purpose of unsupervised object discovery given a single image. TokenCut [120] and Melas-Kyriazi *et al.* [80] both expanded on the work done by LOST [97] which had already improved the state-of-the-art for unsupervised object discovery. The basis of all three works’ methods was their use of DINO features. By applying normalized cuts [93] with a slightly different formulation to the method employed by Melas-Kyriazi *et al.*, TokenCut had the greatest performance out of the three on standard unsupervised object discovery benchmarks which they attribute to the application of the Bilateral Solver [8] as a post-processing step. Extending object discovery to object detection through bounding box assignment has been explored by [100] and [22].

Efforts to begin researching unsupervised deep learning methods of SOD began with [115]. Some examples of unsupervised deep learning which use saliency include [110, 111,

122, 112] and some examples of unsupervised SOD include [60], [140] and TokenCut [120]. Unsupervised SOD can also be considered solved by approaches that solve the more general task of unsupervised semantic segmentation. Unsupervised semantic segmentation methods can be generative, which can involve generating an image segmentation [2], [5], [15], [113] or discriminative which can be done through clustering [54] [53],[145] or contrastive learning[36].

## 2.4 Spectral Methods

Spectral graph theory is the study of a graph’s properties through their matrix representations by examining the eigenvalues, eigenvectors, or characteristic polynomials of those matrices which began with Riemannian manifolds [12]. Fiedler [35] discovered an important result involving the eigenvalues and eigenvectors of the Laplacian matrices of graphs that is an essential component of the method of image segmentation called spectral clustering by showing that the second-smallest eigenvalue of a graph, the Fiedler eigenvalue, corresponds to the algebraic connectivity of the graph. [29] also found interesting mathematical results on using eigenvectors to partition graphs. Building on those results for general machine learning, Ng *et al.* [83] presented a simple standard spectral clustering algorithm involving spectral decomposition of the Laplacian matrix and clustering a select set of its eigenvectors.

In computer vision, Shi and Malik [93] used spectral clustering to do image segmentation by constructing a graph where an image’s pixels are treated as the vertices of a graph and finding a minimum cut that has been “normalized” by dividing by the total edge weight of each partition. Given two disjoint sets of nodes A and B of the nodes V of a graph and  $w(u, v)$ , the edge weight of an edge with endpoints  $u$  and  $v$ , they define a cut as

$$cut(A, B) = \sum_{u \in A, v \in B} w(u, v) \quad (2.3)$$

the total connection of nodes in A to nodes in V as,

$$assoc(A, V) = \sum_{u \in A, t \in V} w(u, t) \quad (2.4)$$

and a normalized cut as

$$Ncut(A, B) = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(A, B)}{assoc(B, V)}. \quad (2.5)$$

Since minimizing normalized cut is NP-complete, they show that the second smallest eigenvector (Fiedler eigenvector) of a Laplacian matrix constructed from the graph gives a good approximate solution.

Finally, in the case of Melas-Kyriazi *et al.* [80], spectral clustering is used to extract semantic segmentations from a specially-constructed affinity matrix by using the fact that DINO features are essentially a downscaled version of the input image. The eigenvector corresponding to the Fiedler eigenvalue could then be used to do SOD. TokenCut [120] also do spectral clustering by applying a normalized cut to a slightly different matrix.

# Chapter 3

## Methods

In this chapter we describe our approach. The goal of this thesis is to develop methods for SOD without supervised data by utilizing DINO [11] features. The advantages of not using labeled data are that it is possible to train on much larger datasets, and there is no inherent bias introduced by human annotators.

The methods we develop here are based on K-means clustering of DINO features. We use two types of K-means clustering: local and global. After clustering, the main problem is to design a criterion for establishing which clusters belong to the salient object, and which clusters belong to the background. We develop several heuristic approaches for this step.

After separating clusters into salient and background, we obtain a rough saliency map, which can be further improved. We use CRF refinement to suppress noise and increase resolution. The final step is to use the resulting refined saliency map as pseudo-ground truth and train a SOD deep learning model on it, to obtain a more accurate SOD classifier. Since our pseudo-ground truth is noisy, we develop an approach which improves the results under noise.

### 3.1 DINO Feature Extraction

Our method begins by preprocessing an input image with a DINO trained ViT and extracting the last layer’s “key” tokens. This choice of features is based on an empirical study that showed a performance comparison of possible feature combinations by LOST [97], which Deep Spectral Methods [80] and TokenCut [120] followed up with by choosing

to use the same features. Each “key” token can be thought of as encoding a 16x16 patch of the input image and is a vector of size 384. If the height or width of the input image is not a multiple of 16, then the image is cropped down to the next multiple of 16 which is done by using floor division. This means that after extracting features, each image is downsampled by a factor of 16.

## 3.2 Local K-Means Clustering

All of our methods are based on K-means clustering. We have two types of clustering, “local” and “global”. In local clustering, described in this section, we cluster DINO features of each image individually. The simplest approach to SOD would be to cluster each image into two clusters ( $K = 2$ ), with the hope that one of the clusters corresponds to the salient object, and another to the background. We performed this experiment, showing the result in Figure 3.1 for some images containing dogs and people as salient objects. The results are unsatisfactory. While the salient object sometimes gets roughly segmented, quite often it is missed, or merged with the background.

The main problem with clustering into two clusters is that there is no guarantee that one of the clusters corresponds to a salient object. A salient object is often contained in a cluster containing other pixels (non-salient) or is broken between the two clusters. This is because DINO features are not extracted using salient object detection training, and they are unlikely to cleanly separate salient and non-salient objects.

Our idea is that if we cluster the image in more than 2 clusters (i.e. perform over-clustering), then it is more likely that some of the resulting clusters correspond to the salient object or significant salient object parts. Consider Figure 3.2. It shows the results of K-means clustering with  $K = 5, 10$  and  $15$ . Note that compared to  $K = 2$ , the salient object is less likely to be grouped with the background parts. From Figure 3.2, we can see that the salient object is often one of the clusters, or a significant part of one of the clusters. The question then becomes how to determine which of the (5,10,15) clusters we should take as the salient and which as the background.



Figure 3.1: K-Means Baseline [80] Initial Investigation Results  
 From left to right - Image, Ground Truth and  $K = 2$  Baseline Prediction

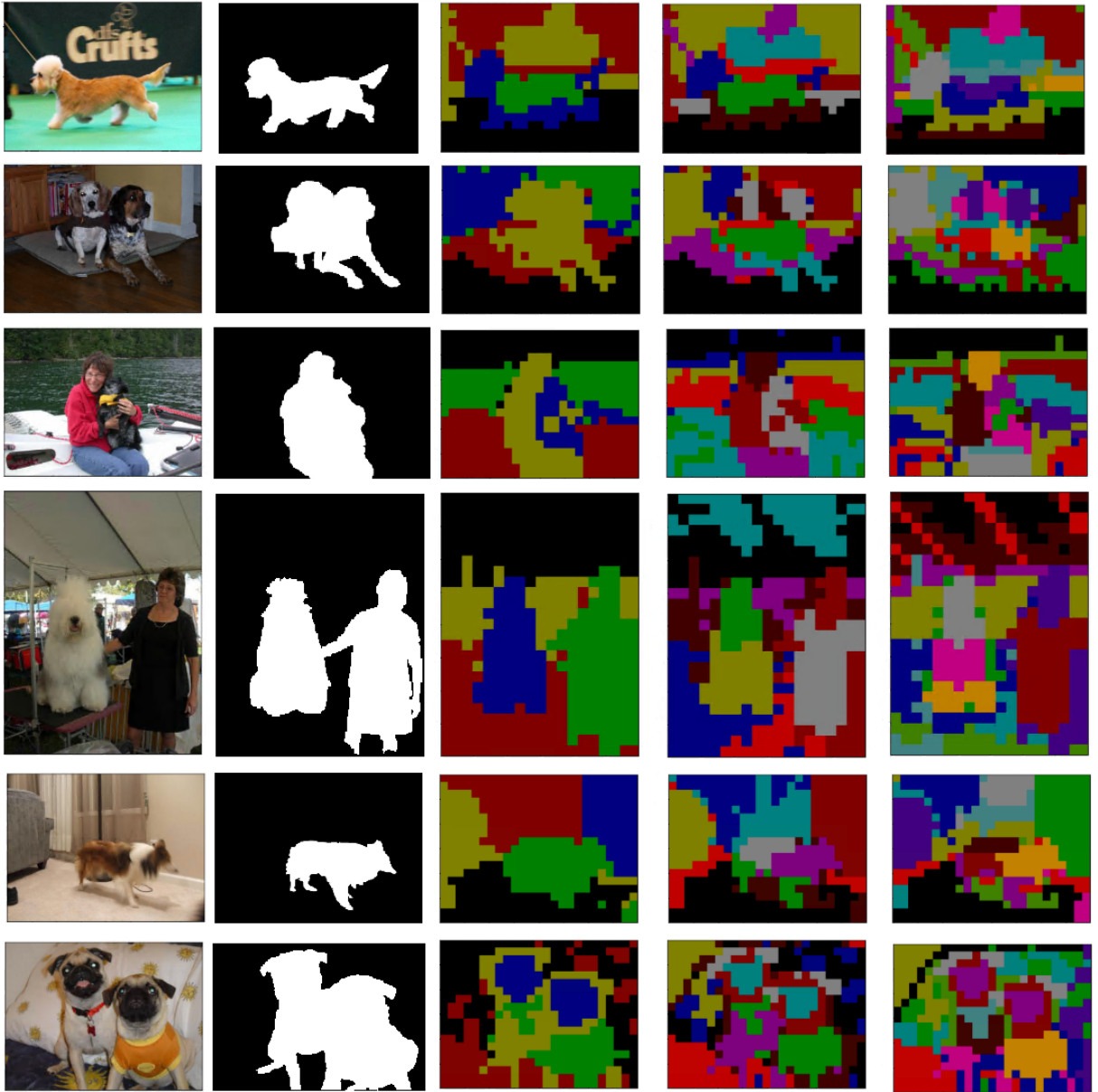


Figure 3.2: Local Clustering Example with  $K = 5, 10$  and  $15$  on the DUTS dataset [115]

### 3.3 Global K-Means Clustering

There are multiple datasets for SOD available for training (please note that we do not use the pixel-precise ground truth that these datasets have). Therefore, we can perform K-means clustering globally, across the whole dataset. The advantage of performing clustering globally is that some salient objects which are split across different clusters in local clustering get assigned to the same cluster. Looking from the point of view of the whole dataset, pixels in these clusters are not as distinct from each other, compared to looking at these pixels from the viewpoint of a single image.

However, performing global clustering across a large dataset is expensive, especially since DINO features are of high dimension. We take the following approach to reduce the computational burden. First, we perform K-means clustering locally (individually for each image), with some value for  $K$ . We refer to the value of  $K$  used for local clustering as local  $K$ . Then we define  $c(f)$  as the cluster label a feature  $f \in \mathbb{R}^{384}$  is assigned to by local K-means clustering. Next, we take cluster centers from each image, and compute the feature vectors corresponding to these local cluster centers by averaging DINO features over pixels inside each local cluster. If the local  $K$  is equal, to say, 5, this gives us 5 feature vectors corresponding to the cluster centers per image. We collect these local cluster center features across all images in the dataset, and perform K-means clustering on all these local cluster centers with a global value of  $K$ , which is usually larger than the local value. After this step, each local cluster center is assigned to a global cluster.

Let  $g(i)$  be the global cluster label assigned to a cluster center whose local cluster label is  $i$ . To perform clustering of all pixels in all images, we perform the following. Let  $c(f)$  be the local cluster label that feature  $f$  is assigned to. Then we assign to feature  $f$  the global cluster label given by  $g(c(f))$ .

Figure 3.3 illustrates some global clustering results for global  $K = 10$ . Focusing on the fifth column we can see that in comparison to the local clustering, the global clustering is able to consolidate the two clusters in the first image that the dog is composed of into one cluster. Global clustering’s ability to identify semantic classes can be seen in the dog and camel in the other two images having the same “red” label as the first image’s dog. These images also have green and blue clusters which could be indicative of common background classes. In particular, green might correspond to walls and blue to the ground/floor.

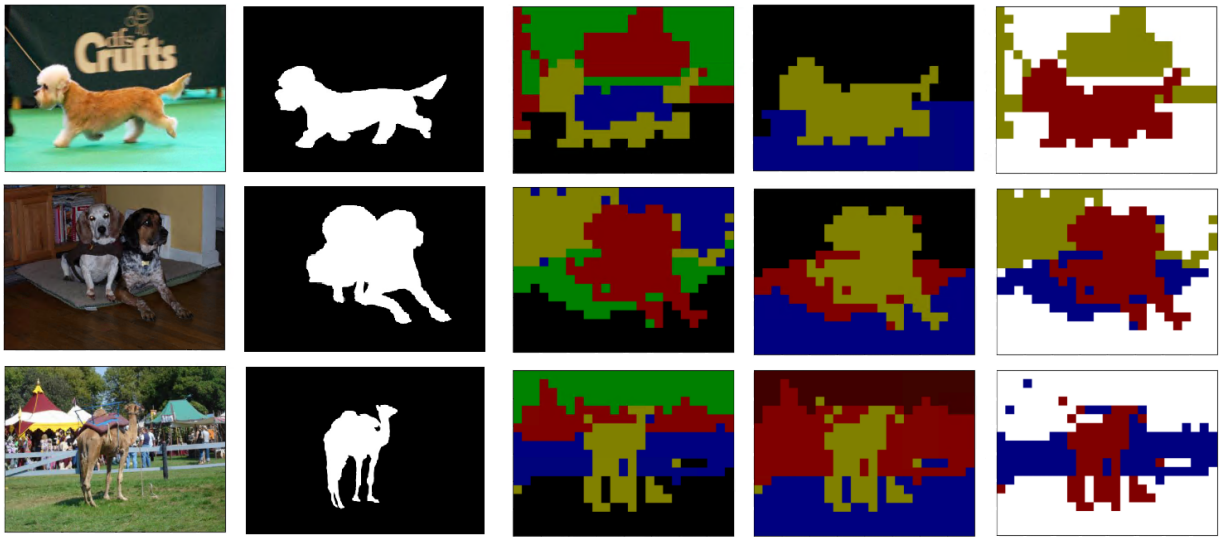


Figure 3.3: Cluster Visualisation. From left to right we have the input image, the ground truth, the clusters produced by the local clustering step, and global clustering. In the fifth column, we show the results of global clustering after unreliable cluster filtering. Unreliable clusters are shown in white. In the first row we can see that the two clusters originally comprising the dog were merged by the global clustering step.

### 3.4 Unreliable Cluster Filtering

Our next step is filtering out unreliable clusters. Unreliable cluster filtering is performed on the results of local clustering. But it affects global clustering as well, since global clustering is based on the results of local clustering. A cluster that corresponds to some frequent salient objects, such as dogs, is likely to have a similar cluster across many other images. Therefore, a cluster is deemed reliable if there are many clusters that are similar to it in other images.

Intuitively, two clusters are similar, if their centers are close to each other. But the closeness of their centers is best judged in the relation to the compactness of points in each cluster, see Figure 3.4. This leads to an intuitive definition of two clusters as similar if the distance between their centers is less than the compactness of both.

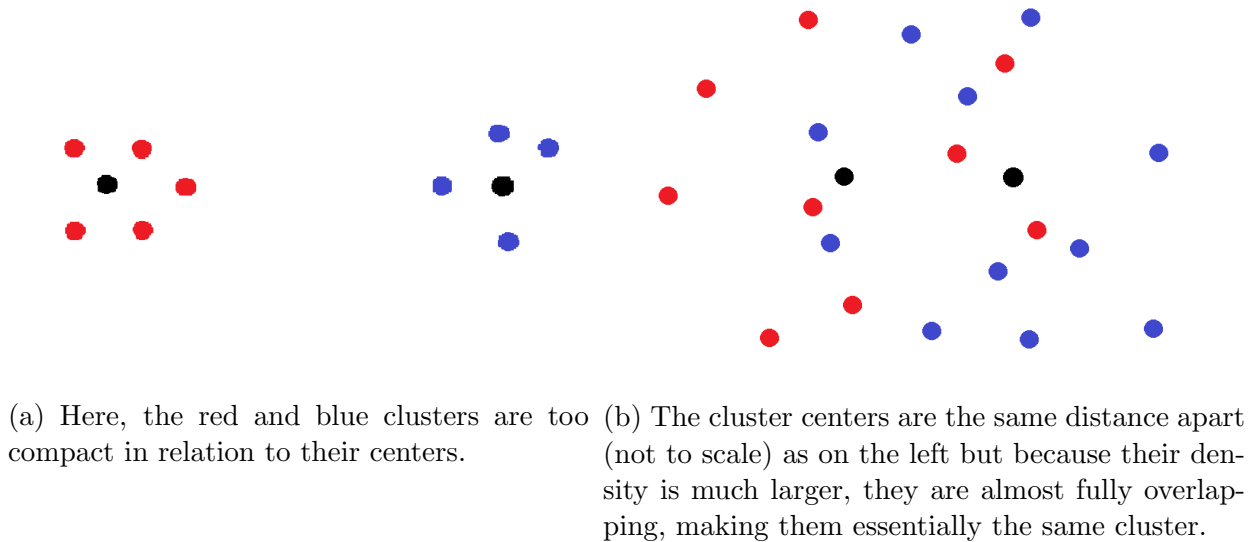


Figure 3.4: Illustration of our criterion for cluster similarity. The points in two clusters are shown with red and blue. Cluster centers are shown with black.

We define compactness as the average distance of the points in the cluster to the cluster center. Let  $x \in \mathbb{R}^{384}$  be the feature vector corresponding to a local cluster center for an image  $I$  and cluster  $C_x$  be the set of features in image  $I$  that belong to the cluster with center  $x$ , i.e.  $C_x = \{f \in I \mid c(f) = c(x)\}$ . Then the compactness of cluster  $C_x$  is defined

as,

$$compactness(C_x) = \frac{\sum_{f \in C_x} \|f - x\|_2}{|C_x|} \quad (3.1)$$

Next we calculate cluster similarity counts. This is done for each cluster across the dataset. We keep the count of the number of clusters similar to the cluster  $C_x$  in the variable  $similarity\_count(C_x)$ . If the distance between a pair of cluster centers  $(x, y)$  is less than both cluster compactnesses ( $compactness(C_x)$ ,  $compactness(C_y)$ ), then both of their clusters ( $C_x$ ,  $C_y$ ) are given 1 to their similarity counts ( $similarity\_count(C_x)$ ,  $similarity\_count(C_y)$ ) and 0 otherwise. In pseudocode this decision can be described by Algorithm 1. The only cluster pairs that are not considered are the ones where both clusters belong to the same image. This means for a set of 500 images with  $K = 5$  for local clustering, each of the 2500 clusters can have a similarity count of up to  $(500 - 1) \times 5 = 2495$ .

---

**Algorithm 1** Computing similarity counts

---

**Input:**  $D$  is the full set of local cluster centers

```

for all  $x \in D$  do
  for all  $y \in D$  |  $y$  is not in the same image as  $x$  do
     $center\_dist \leftarrow \sqrt{\sum_{i=1}^{384} (x_i - y_i)^2}$ 
    if  $center\_dist < compactness(C_x)$  and  $center\_dist < compactness(C_y)$  then
       $similarity\_count(C_x) \leftarrow similarity\_count(C_x) + 1$ 
    end if
  end for
end for

```

---

We now determine which clusters are reliable. If a cluster  $C_x$  has  $similarity\_count(C_x) \geq 2\%$  (total number of clusters), then  $C_x$  is considered reliable.

Before applying the “global” clustering step we can make a decision to prefilter clusters that were deemed unreliable (their centers are not considered for K-means). Intuitively, unreliable clusters do not occur frequently throughout the dataset and are unlikely to correspond to salient objects. Therefore, we label any unreliable clusters as background.

## 3.5 Salient Cluster Determination: Local Heuristics

After performing the local and global clustering steps and filtering out the unreliable clusters, we are ready for the task of determining which cluster or clusters belong to the salient object, and which to the background. In this section, we describe heuristics based on a single image, i.e. local heuristics.

### 3.5.1 Center Prior

Taking note of the center prior, which is the tendency for a salient object to be in the center of an image, we assign clusters that do not touch the image border to be salient, and the rest of the clusters we assign to the background. A variation of this heuristic is to assign the clusters that do not touch the corners of the image to the salient object, and the rest to the background. This variation did not have as strong results as assigning clusters that touch the image border to the background, see Chapter 4.

See Figure 3.5 for some qualitative center prior illustrations. The border-based heuristic works better for the second and third row, while the corner heuristic works better for the first row.

Figure 3.6 illustrates how unreliable cluster filtering helps to obtain more robust results with our center prior. As shown in the two example input images, unreliable cluster filtering is particularly useful for images which happen to have a “border”. Such a border gets assigned to a separate cluster. Without unreliable cluster filtering, such “border” clusters get assigned to the background and the rest of the clusters to the salient object. With unreliable cluster filtering, the unreliable clusters (blue and red in the top row, and green and yellow in the bottom row) also get assigned to the background, resulting in much more accurate masks.

### 3.5.2 Least Border

This heuristic is used in conjunction with the Center Prior heuristic described in Section 3.5.1. After using the Center Prior heuristic, some images may have no salient pixels, i.e. everything is assigned to the background. However, for the training dataset, we know that each image has a salient object. In order to attempt to rectify the absence of a salient object, while still using the center prior inspired approach, we find the cluster that touches the image border the least and assign it to the salient object. Figure 3.7 illustrates

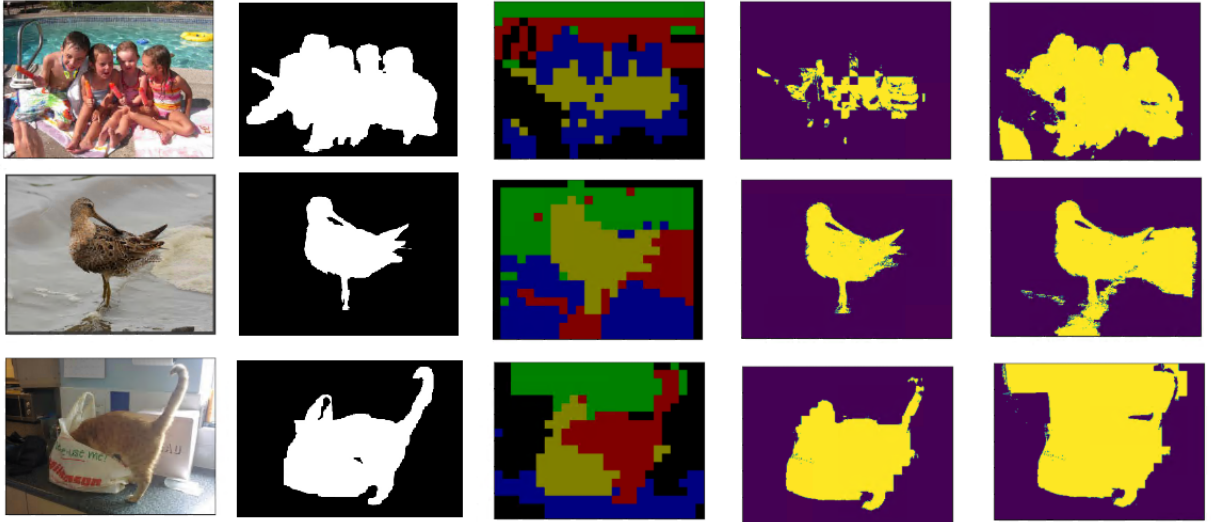


Figure 3.5: From left to right - Image, Ground Truth, Local Clustering, Center prior (border-based), Center Prior (corner-based)

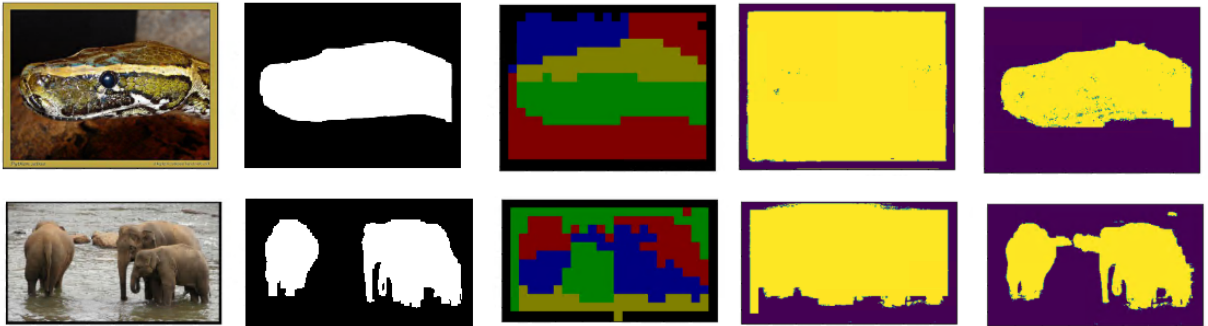


Figure 3.6: From left to right - Image, Ground Truth, Local Clustering, Without Unreliable Cluster Filtering, Unreliable Cluster Filtering

some cases where the least border heuristic helps. In these images, all clusters touch the image border, and therefore all pixels are labeled as background under the “Center Prior” heuristic. But with the least border heuristic, at least some clusters are labeled as salient.

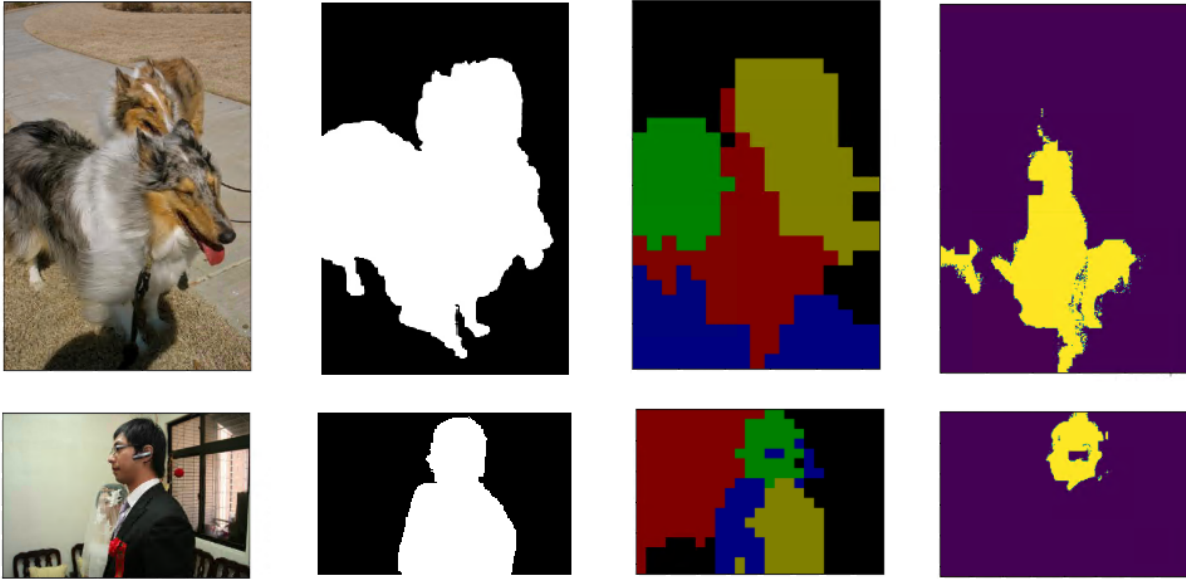


Figure 3.7: From left to right - Image, Ground Truth, Local Clustering, Least Border Heuristic

### 3.5.3 Local Spatial Centeredness

This heuristic is also based on the fact that the salient object is more likely to be in the center of the image. As opposed to the heuristic in 3.5.1, where we take all clusters not touching the image's border (or corners) to be salient, here we want to measure the centeredness, and chose salient clusters based on this measure. We will define the centeredness of a cluster to be the average spatial distance from pixels in the cluster to the center of the image. With this heuristic, a cluster can be the most centered but still touch an image border, and, therefore, be selected for saliency.

#### Bottom $k$

We first tried assigning the bottom  $k$  (closest to the image center) spatially centered clusters to be salient.

## Enumeration Selection

Another way to assign saliency according to centeredness is to enumerate all possible assignments of the local clusters into two segments for a total of  $2^K$  assignments. Each assignment’s two segments can have their centeredness computed, but it’s not so clear how to proceed and choose the best assignment. One way we have explored is to choose the assignment with the largest difference between the centeredness of the two segments.

## Adaptable Selection

This heuristic selects the salient clusters adaptively, one by one. We start by assigning the cluster which is closest to the image center to be salient. We assume that this cluster is most likely to be salient, and therefore, the center of this cluster should be the new “center” with respect to which we should do all the following computations. Therefore, the next cluster to be chosen as salient is the cluster which is closest to this newly computed center. After we add the second cluster, we recompute the “center” as the spatial average of the pixels in these two clusters. We proceed by adding clusters to be salient in this manner, recomputing the “center” after each new addition, until  $k$  clusters have been selected.

### 3.5.4 Color Histogram

Our next heuristic is based on the idea that the salient object and the background should have a large difference in appearance. To model the appearance of a cluster, we use normalized color histograms. Since the number of colors is high, to avoid sparsity problems and for efficiency, we quantize color into bins (we tried 3, 8, 16, and 32 bins).

Given two segments, to compute the difference between their appearance, we compute their normalized color histograms and then measure either the L1, L2 norms or the KL-divergence between these histograms.

We first divide all clusters into two segments, and then determine which segment is the salient object, and which is the background.

Our first approach is to choose the most distinct single cluster as one segment. To do this, we iterate over all clusters, compute the difference in their histograms, and then take the cluster leading to the largest difference as one segment, and the rest of the clusters as another segment.

Our second approach is to go over all possible segment memberships and choose the one giving the largest histogram difference. Suppose we have  $K = 5$  clusters in the image. Then we have  $2^5 = 32$  possible assignments of these clusters to 2 segments. We enumerate all possible assignments, compute the histogram difference for each, and chose the segmentation giving the largest histogram difference.

Finally, given 2 segments (produced by the first, single cluster method or the second, considering all possible memberships), we assign to the salient object the cluster which has the best spatial centeredness.

### 3.6 Salient Object Determination: Global Heuristics

Previous heuristics made decisions about the saliency of a cluster for each image individually. We also consider heuristics which are based on the results of global clustering. These heuristics make a decision about a global cluster as a whole. If some global cluster is determined to be the salient object, then for all images which have that cluster, the pixels in that cluster are assigned to the salient object. Each global cluster has pixels in individual images, and the set of pixels from a cluster in an individual image is called a local cluster. Below we explore 4 global cluster based heuristics.

#### Global Border Ratio 1

The center prior is useful for salient cluster determination. Here, we want to apply the center prior to an entire global cluster. That is, we want to measure, on average, how often a global cluster touches an image’s border.

A global cluster  $G$  has pixels spread around multiple images. Let  $I$  be an image that contains pixels from cluster  $G$ . Then we will denote the set of those pixels as the local cluster  $G_I$ . We count how many of these pixels touch the border of image  $I$ , and then divide by the number of pixels from the cluster,  $|G_I|$ . This gives us a measure of how much a cluster touches the image border for one individual image. We will call this measure  $borderRatio1(G_I)$  for local cluster  $G_I$ .

Finally, we add up  $borderRatio1(G_I)$  over all images that have pixels from cluster  $G$  and divide by the total number of such images. A global cluster that does not touch border in any of the images has the best possible score according to this heuristic. We choose as salient, the top  $K$  clusters according to this heuristic.

## Global Border Ratio 2

In this heuristic we do every calculation the same as in Global Border Ratio 1 but instead of dividing by the number of pixels that make up the local cluster,  $|G_I|$ , we divide by the number of pixels that make up the image border. That is to say, if  $I$  has height  $H$  and width  $W$ , then border  $borderRatio2(G_I)$  is computed by dividing by  $2(H + W) - 4$ .

This variant of the heuristic is tuned more towards how much of an image border a cluster touches, whereas the heuristic in the previous section is tuned more to what portion of a cluster touches the image border.

## Global Border Count

This heuristic counts how many local clusters belonging to a global cluster touch the border. Global clusters with lower border counts are considered better.

## Spatial Centeredness

The idea here is again, that the salient objects are more likely to be in the image center. However, here we measure the spatial centeredness explicitly, just as we did for local clustering in Section 3.5.3. However, here we do a global calculation. We calculate spatial centeredness of a global cluster by computing the average distance of each pixel  $p$  in the cluster to the center of the image pixel  $p$  actually belongs to. In the end, the global clusters with better local centeredness are considered better which leads us to take the bottom (closest to the center) several global clusters to be salient.

## 3.7 Post-processing

### Size Filter

We expect salient objects to be at least 2% of the image size, so images where salient objects are smaller than that are likely erroneous. We use our approach to construct pseudo-ground truth for training an SOD model. It is desirable to minimize errors in pseudo-ground truth. Therefore, we remove from the training dataset all images where the salient object occupies only a very small portion. In practice, we remove images where

the salient object is less than or equal to 2% of the image size. This results in a smaller dataset, but with more accurate pseudo-ground truth.

## Applying a Conditional Random Field (CRF)

The final post-processing step is to upsample the coarse saliency masks to the original image size using nearest neighbour interpolation and applying the dense CRF developed by Krähenbühl *et al.* [62]. Upsampling from 16x16 patches gives rough squares and applying a CRF can smoothen the segmentation.

In computer vision, a conditional random field is usually defined by treating the pixels of an input image  $I$  and a set of labels for those pixels  $X$  as random fields so that the probability of  $X$  can be conditioned on  $I$ . For the dense CRF we use, the goal is to find a set of binary labels  $x$ , where 0 corresponds to the background and 1 corresponds to the salient object, that minimizes the following objective function (Gibbs energy),

$$E(x) = \sum_{x_i} u(x_i) + \sum_{x_i, x_j} v(x_i, x_j), \quad (3.2)$$

where  $x_i, x_j \in x$  are the labels for pixels  $i, j$  and  $u(x_i)$  and  $v(x_i, x_j)$  are functions known as the unary potential and pairwise potential.

The unary potential should act on individual labels and takes into account the saliency mask while the pairwise potential should act on all possible pairs of pixels and considers the input image. In particular, the unary potential can be computed as,

$$u(x_i) = \begin{cases} -\ln(0.99), & x_i = y_i \\ -\ln(0.01), & \text{otherwise} \end{cases} \quad (3.3)$$

where  $y_i$  is the label that the saliency mask gives to pixel  $i$ .

The pairwise potential for this dense CRF is based on a linear combination of Gaussian kernels,

$$v(x_i, x_j) = [x_i \neq x_j] \left( w_1 e^{-\left(\frac{|p_i - p_j|^2}{2\alpha^2} + \frac{|I_i - I_j|^2}{2\beta^2}\right)} + w_2 e^{-\frac{|p_i - p_j|^2}{2\gamma^2}} \right) \quad (3.4)$$

where  $[.]$  is the Iverson bracket and  $p_i, p_j$  and  $I_i, I_j$  are the spatial coordinates and color vectors respectively for pixels  $i, j$ .

To solve this problem efficiently, an algorithm that finds an approximate solution was developed that runs for a number of iterations which we set to 5. The rest of the parameters are set as  $w_1 = 10, \alpha = 80, \beta = 13, w_2 = 3, \gamma = 3$ .

See Figure 3.8 for some examples of segmentations before and after the application of this CRF.

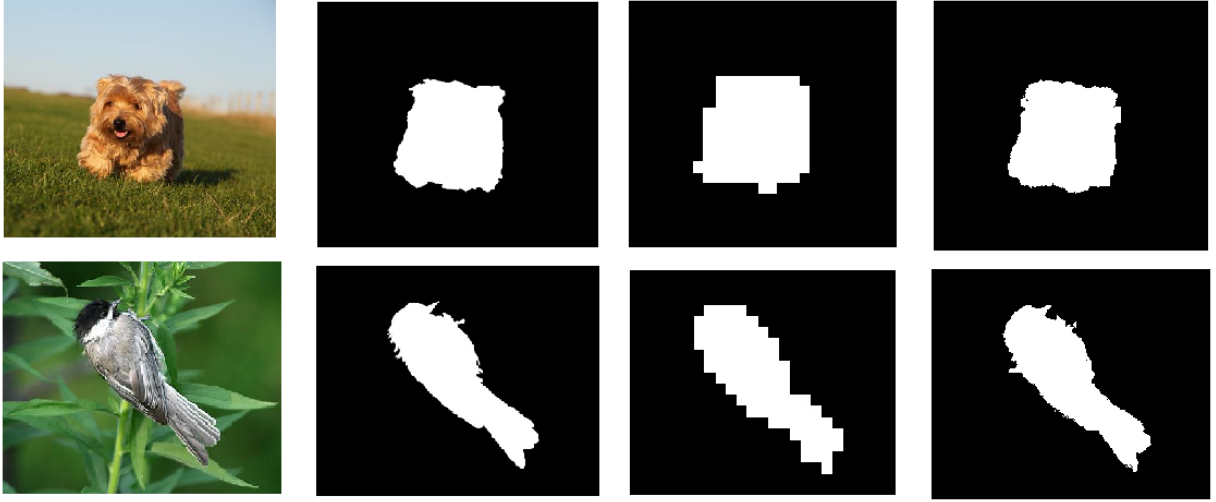


Figure 3.8: From left to right - Image, Ground Truth, No Crf, With CRF

### 3.8 Training a Supervised Network

The last step of our approach is to use the obtained saliency masks as pseudo-ground truth and to train a neural network using this pseudo-ground truth for supervision. Specifically, we use SelfReformer [136] as it is a recent SOD neural network which produces state-of-the-art results when used for SOD with full supervision. Our main method of labeling clusters as salient, the Center Prior, results in some images which do not have any clusters labeled as salient. We exclude such images from training, as they are clearly erroneous (every pixel cannot be background in a SOD dataset). Other methods, such as Local Spatial Centeredness, do not have this disadvantage, that is they always produce images with some cluster labeled as salient.

## Noisy Pseudo-ground Truth

Our pseudo-ground truth is noisy, see Figure 3.9. In particular, we find that the pixels labeled as salient are more likely to be salient than the pixels that are labeled as background. As the salient pixels are more reliable than the background pixels, we decided to keep the salient pixels as is in our pseudo-ground truth, but use only some of the pixels labeled as the background for training. We replace pixels in the background which are not used for training with a “void” label. A void label is ignored during supervised training.

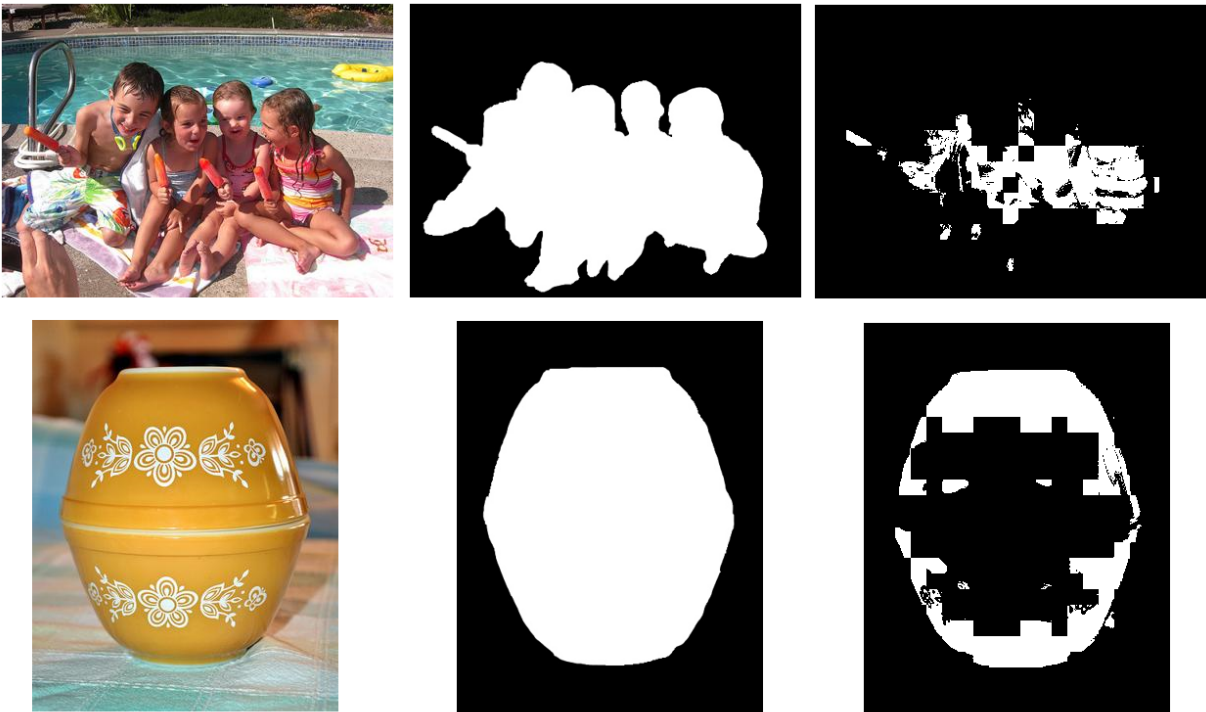


Figure 3.9: From left to right - Image, Ground Truth, Pseudo-ground Truth

If every background pixel is set to void, then the supervision will lead to the degenerate fully salient solution, so it is not a plausible way to implement void pixels. One plausible way to implement void pixels is to set every background pixel to be void except for a border of 3-5 pixels in width, as the image border is most likely to be the background. However, experimental results with this method were quite poor, perhaps because there were no viable examples of background pixels which are not on the image border. A second way to implement void pixels is to assign some percentage of the pixels that are not the object to

be the background. A third way is to label the border of the object as background which can be done by applying the morphological operation “dilate”. Empirical analysis shows that labeling a random subset of pixels as void works the best. In particular, we label 80% of random background pixels as void. See Figure 3.10 for an example of the different ways to assign void pixels.



Figure 3.10: From left to right - The Original Image, Ground Truth, Object Border Background and 80% Random Void

## Using a Pretrained Backbone

We use the SelfReformer with a pretrained PVTv2 backbone [117] as originally intended by the authors of SelfReformer which means that before training with our pseudo-ground truths the SelfReformer has undergone image-level supervision. This means that any results obtained at specifically this point cannot be classified as unsupervised, but they still achieve the goal of learning without pixel-level annotations. See Figure 3.11 for a comparison of the predictions made by a SelfReformer after 200 epochs of training with and without pretraining.



Figure 3.11: From left to right - Input Image, Ground Truth, Prediction without pretraining, Prediction with pretraining

# Chapter 4

## Experiments

In this chapter we report the results of the experiments we conducted of our approach as well as provide an analysis of the reported results. We begin by outlining the dataset, evaluation metric and training protocols used. Then, we discuss the accuracy of our pseudo-ground truth, the accuracy of the predictions made by a SelfReformer trained on that pseudo-ground truth as well as some ablation experiments of our proposed heuristics. Finally we show some qualitative results.

### 4.1 Benchmark Dataset

We use the DUTS dataset [115] in this thesis. DUTS is one of the largest SOD datasets. It is divided into the DUTSTR and DUTSTE subsets, containing 10553 and 5019 images respectively. These images have been sampled from ImageNet [26] and SUN [128]. In particular, DUTSTR contains a large proportion of single object images that can be solved more easily making it a relatively easier benchmark compared to DUTSTE which is almost completely composed of images depicting complex scenes. DUTSTR is typically used for training, and DUTSTE for evaluation. We train SelfReformer on pseudo-ground truth obtained on DUTSTR, and evaluate the performance on DUTSTE.

## 4.2 Evaluation Metrics

### Mean Absolute Error (MAE)

One way to evaluate performance in SOD is to use the Mean Absolute Error (MAE) metric which counts the number of wrong predictions and divides by the total number of predictions. If  $P \in [0, 1]^{HW}$  and  $T \in [0, 1]^{HW}$  are the prediction and target saliency masks, then MAE can be computed using

$$MAE = \frac{\sum_i |P_i - T_i|}{HW}. \quad (4.1)$$

### F-Measure

Another metric for evaluating SOD performance is the  $max\_F_\beta$  metric which is based on the F-Measure metric. The  $max\_F_\beta$  metric is the most common metric for evaluating the performance of an SOD method. It is defined as below.

If an input image has height  $H \in \mathbb{Z}$  and width  $W \in \mathbb{Z}$ , then let  $P \in [0, 255]^{HW}$  and  $T \in \{0, 1\}^{HW}$  be the prediction and target saliency masks.  $P$  stores a score from 0 to 255 of how likely it is for a pixel to belong to a salient object, with 255 indicating most likely. For most applications, one needs to threshold  $P$  to produce a binary answer, whether each pixel is salient or not. However, it is not entirely obvious which threshold is the best. The idea behind  $max\_F_\beta$  is that instead of some fixed threshold, we are going to evaluate over all possible thresholds and choose the best one.

We threshold  $P$  at the 255 possible  $x \in [0, 255)$  to convert it to a binary saliency mask  $P_x \in \{0, 1\}^{HW}$  such that  $P_x$  is 1 wherever  $P > x$  and 0 otherwise. This allows us to compute the “precision” and “recall” of  $P$  against  $T$  for every image as

$$precision = \frac{\sum_{z \in Z} z}{\sum_{p \in P_x} p} \quad (4.2)$$

and

$$recall = \frac{\sum_{z \in Z} z}{\sum_{t \in T} t}. \quad (4.3)$$

where  $Z \in \{0, 1\}^{HW}$  is 1 wherever  $P_x$  and  $T$  are both 1 and 0 otherwise. In words, precision is the number of positive predictions that are correct while recall is the number of positive targets predicted correctly. The result is that both precision and recall are calculated for each image of the evaluation dataset for all 255 thresholds.

The traditional  $F_\beta$  metric can then be calculated as

$$F_\beta = (1 + \beta^2) \frac{\text{precision} \times \text{recall}}{(\beta^2)\text{precision} + \text{recall}} \quad (4.4)$$

where  $\beta$  is a hyperparameter commonly set to 0.3.  $\beta$  measures how important recall is (a large  $\beta$  means that you would rather have a worse precision in the denominator), so a  $\beta$  value of 0.3 means recall is only 0.3 times as important as precision. This is good for not rewarding the degenerate solution that makes positive predictions for every pixel. Finally, we take the maximum average  $F_\beta$  (over every image) for each threshold  $x$  to get  $\max F_\beta$ ,

$$\max F_\beta = \max_{x \in [0, 255]} \text{average}_x(F_\beta) \quad (4.5)$$

where  $\text{average}_x(F_\beta)$  denotes the average  $F_\beta$  for the threshold  $x$ . Without  $\beta$ , the traditional F-Measure would be computed as the harmonic mean of precision and recall,

$$\max F_\beta = 2 \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad (4.6)$$

which is essentially a special case of  $F_\beta$  when  $\beta = 1$ . An F-Measure-based metric is better suited for evaluation of SOD, compared to the MAE metric, in datasets where the salient object size tends to be much smaller than the background. For such datasets, if we have a predictor which produces a degenerate solution by labeling every pixel as the background, since the number of salient pixels is much smaller compared to the background, MAE will actually give a deceptively good (relatively low) score. However, the F-measure-based score will be 0, which is the worst possible score.

## 4.3 SelfReformer Training

### Our Main Method

Our main method begins with  $K = 5$  local K-means clustering on DUTSTR. Then we apply unreliable cluster filtering and the border-based center prior heuristic to choose

salient clusters. Finally we remove the saliency masks that take up 2% or less of the total image size and apply a CRF. The remaining masks have their performance evaluated before moving on to the training of the SelfReformer which will use those masks as pseudo-ground truth. One possible improvement to the training is the choice of loss function.

## Choice of Loss Function

We test some loss function alternatives against the weighted binary cross entropy (BCE) that SelfReformer uses such as standard BCE and collision cross entropy (CCE) [146].

### Binary Cross Entropy (BCE)

Given a single prediction  $x \in [0, 1]$  for a target  $y \in \{0, 1\}$ , BCE can be computed as

$$BCE(x, y) = -(y \ln(x) + (1 - y) \ln(1 - x)). \quad (4.7)$$

The closer  $x$  is to  $y$  the closer BCE is to zero loss and on the flipside, the farther  $x$  is away from  $y$ , the closer BCE is to an infinite amount of loss. We compute BCE on the final output of SelfReformer.

### Loss Function in SelfReformer

The original loss function SelfReformer uses is a form of weighted BCE which can be summarized as a linear combination of a modified BCE known as Pixel Position Accuracy (PPA) [121] applied to 9 separate features from throughout the network. Let  $P^i, 1 \leq i \leq 9$  be the 9 output features that SelfReformer produces in order of increasing depth so that  $P^1$  denotes the shallowest features and  $P^9$  the deepest features (which is used for prediction) and the target be  $G$ . Then PPA can be computed as,

$$PPA(P^i, G) = \frac{\sum_j (1 + \gamma \alpha_j) BCE(P_j^i, G_j)}{\sum_j \gamma \alpha_j} \quad (4.8)$$

where  $\alpha_j$  is computed as,

$$\alpha_j = \left| \frac{\sum_{k \in A_j} G_k}{m \times n} - G_j \right|. \quad (4.9)$$

where  $A_j$  is a surrounding patch of size  $m \times n$  around pixel  $j$ . The hyperparameters  $\gamma$ ,  $m$  and  $n$  were set to 4, 31 and 31 in SelfReformer. Every second segment,  $P^i \in \{P^3, P^5, P^7, P^9\}$  also has its Confident Loss per pixel computed as,

$$\text{Confident}(P_j^i, G_j) = \frac{2 \times G_j \times (1 - G_j) \times (\text{BCE}(P_j^i, G_j)) + \lambda}{|G|}. \quad (4.10)$$

where  $\lambda$  is set to  $\frac{1}{2}$  by default. Finally, the total loss can be computed by summing together a linear combination of PPA on  $P^i, 2 \leq i \leq 9$  and the 4 Confident Losses with standard BCE on  $P^1$  (which was produced by the global context branch). This loss function is good for consolidating the information produced by the global context branch and the multi-scale features so that SelfReformer can make more detailed predictions.

### Collision Cross Entropy

In [146], they introduce Collision Cross Entropy (CCE), which has an advantage over the standard BCE whenever labels have noise, which is the case with our pseudo-ground truth. It is based on modeling the probability that two random variables  $x$  and  $y$  are equal. CCE is computed as

$$\text{CCE}(x, y) = -\ln(y_1 x_1 + y_2 x_2). \quad (4.11)$$

To adapt our saliency masks which are hard labels in  $\{0, 1\}$  for CCE we convert them to a distribution over labels, i.e. to soft labels, such as  $\{0.2, 0.8\}$ ,  $\{0.3, 0.7\}$  or  $\{0.4, 0.6\}$ . It's important to not have the soft label corresponding to 0 be greater than the one corresponding to 1 when using CCE, because maximizing the inside of the  $\ln$  is effectively equivalent to minimizing the loss.

Just as in BCE, we compute CCE solely on the final predictive features from SelfReformer, i.e. on  $P^9$ .

## 4.4 Pseudo-Ground Truth Accuracy

In this section, we compare the accuracy of pseudo-ground truth generated with our main method and the prior work method we compare to, eigenvectors [80]. The accuracy of pseudo-ground truth affects the final results, i.e. the accuracy of the SelfReformer training.

## Comparison of our Main Method to Eigenvectors

We first compare the main method to the eigenvectors produced by [80] in Table 4.1. The eigenvector’s performance is already quite good but if we adjust them spatially by reassigning the clusters according to local spatial centeredness we can improve it even further on average. To do this, we compute the centeredness of the two segments as described in Chapter 3 and swap their labels if the segment which is closer to the center was originally labeled as background.

|                                   | $\max F_\beta$ | MAE    | Set Size |
|-----------------------------------|----------------|--------|----------|
| Main Method                       | 0.8028         | 0.1209 | 7097     |
| Main Method (Size Filter)         | 0.8104         | 0.1199 | 6989     |
| Eigenvectors                      | 0.8114         | 0.1255 | 10553    |
| Eigenvectors (Spatially adjusted) | 0.8712         | 0.0707 | 10553    |

Table 4.1: The accuracy of pseudo-ground truth of our main methods compared to the eigenvectors [80] method, and eigenvectors [80] methods with spatially adjusted salient regions. Set size is the number of training images. Our method discards some of the training images which results in a smaller training set.

## Main Method

### Varying Local K-means Clustering

Our main method uses local clustering with  $K = 5$ . In Table 4.2, we evaluate the accuracy of pseudo-ground truth with other values for local  $K$ . For  $K = 2$  the accuracy is much higher, and for  $K = 10$  the accuracy is much lower. However, the training set size is much smaller with  $K = 2$ , so the results of training SelfReformer with  $K = 2$  (see Section 4.6 and Table 4.12) are actually worse compared to  $K = 5$ .

### Varying Global K-means Clustering

Our main method does not use global clustering but we still compare different values of global  $K$ . In Table 4.3, we evaluate the accuracy of pseudo-ground truth with other values for global  $K$ . For  $K = 5$  the accuracy is marginally the best, and for  $K = 50$  the accuracy is slightly lower.

|                      | $max F_\beta$ | MAE    | Set Size |
|----------------------|---------------|--------|----------|
| K = 2                | 0.9239        | 0.0377 | 2758     |
| K = 2 (Size Filter)  | 0.9242        | 0.0377 | 2756     |
| K = 5                | 0.8028        | 0.1209 | 7097     |
| K = 5 (Size Filter)  | 0.8104        | 0.1199 | 6989     |
| K = 10               | 0.5700        | 0.1998 | 7714     |
| K = 10 (Size Filter) | 0.6464        | 0.1887 | 6543     |

Table 4.2: Performance comparison of local K = 2, 5 and 10 on the training set, DUTSTR.

|                           | $max F_\beta$ | MAE    | Set Size |
|---------------------------|---------------|--------|----------|
| Main Method               | 0.8028        | 0.1209 | 7097     |
| Main Method (Size Filter) | 0.8104        | 0.1199 | 6989     |
| K = 5                     | 0.8055        | 0.1222 | 6050     |
| K = 5 (Size Filter)       | 0.8153        | 0.1207 | 5938     |
| K = 10                    | 0.8045        | 0.1251 | 6426     |
| K = 10 (Size Filter)      | 0.8135        | 0.1237 | 6316     |
| K = 20                    | 0.8003        | 0.1282 | 6797     |
| K = 20 (Size Filter)      | 0.8085        | 0.1272 | 6688     |
| K = 50                    | 0.7988        | 0.1299 | 6997     |
| K = 50 (Size Filter)      | 0.8070        | 0.1289 | 6885     |

Table 4.3: Performance comparison of global K = 5, 10, 20 and 50 on the training set, DUTSTR.

## Other Variants of our Main Method

In Table 4.4 we can see the accuracy of the pseudo-ground truth of our main method compared to its other variations.

See Table 4.5 for the accuracy of the ground truth used in our ablations. Unreliable filtering helped the local spatial centeredness method but it didn't help the main method for K = 5.

|                           | $max\_F_\beta$ | MAE    | Set Size |
|---------------------------|----------------|--------|----------|
| Main Method               | 0.8028         | 0.1209 | 7097     |
| Main Method (Size Filter) | 0.8104         | 0.1199 | 6989     |
| LSC                       | 0.7317         | 0.1544 | 10539    |
| LSC (Size Filter)         | 0.7426         | 0.1532 | 10301    |
| LSCU                      | 0.7669         | 0.1379 | 7529     |
| LSCU (Size Filter)        | 0.7742         | 0.1370 | 7412     |
| GSC                       | 0.7287         | 0.1242 | 885      |
| GSC (Size Filter)         | 0.7353         | 0.1244 | 858      |
| GBR1                      | 0.7535         | 0.1270 | 799      |
| GBR1 (Size Filter)        | 0.7591         | 0.1277 | 783      |

Table 4.4: Main method variations performance comparison. LSC: Local spatial centeredness from Section 3.5.3. LSCU: Local spatial centeredness with unreliable cluster filtering from Sections 3.5.3 and 3.4 respectively. GSC: Spatial Centeredness Global Heuristic from Section 3.6. GBR1: Global Border Ratio 1 from Section 3.6.

|                                               | $max\_F_\beta$ | MAE    | Set Size |
|-----------------------------------------------|----------------|--------|----------|
| Main Method                                   | 0.8028         | 0.1209 | 7097     |
| Main Method (Size Filter)                     | 0.8104         | 0.1199 | 6989     |
| No Unreliable Cluster Filtering               | 0.8295         | 0.1047 | 8686     |
| No Unreliable Cluster Filtering (Size Filter) | 0.8339         | 0.1044 | 8594     |
| No CRF                                        | 0.7727         | 0.1364 | 7102     |
| Corner Cut                                    | 0.7197         | 0.1562 | 9938     |
| Corner Cut (Size Filter)                      | 0.7250         | 0.1558 | 9833     |

Table 4.5: Performance comparison of ablation ground truth. The 2% size filter did not change the ground truth when not using a CRF.

## 4.5 SelfReformer Training Results

The best performing methods results are shown in Table 4.6. All predictions are on the DUTSTE dataset obtained after 200 epochs of training on the DUTSTR dataset. Some training baselines are shown in Tables 4.7 and 4.8. In Table 4.9, we compare the results of our main method to the more complicated method based on eigenvectors [80]. Surprisingly, our performance is better. For eigenvectors, we also take the pseudo-ground truth they produce and train on SelfReformer for 200 epochs.

|                           | $max\_F_\beta$ | MAE    |
|---------------------------|----------------|--------|
| Main Method               | 0.7593         | 0.0776 |
| Main Method (Size Filter) | 0.7670         | 0.0767 |
| LSC                       | 0.6916         | 0.0965 |
| LSC (Size Filter)         | 0.6915         | 0.0981 |
| LSCU                      | 0.7200         | 0.0870 |
| LSCU (Size Filter)        | 0.7217         | 0.0878 |
| GSC                       | 0.6289         | 0.1141 |
| GSC (Size Filter)         | 0.5908         | 0.1334 |
| GBR1                      | 0.6486         | 0.1127 |
| GBR1 (Size Filter)        | 0.6435         | 0.1142 |

Table 4.6: Post-training performance comparison for main method variations. LSC: Local spatial centeredness from Section 3.5.3. LSCU: Local spatial centeredness with unreliable cluster filtering from Sections 3.5.3 and 3.4 respectively. GSC: Spatial Centeredness Global Heuristic from Section 3.6. GBR1: Global Border Ratio 1 from Section 3.6.

|                           | $max\_F_\beta$ | MAE    |
|---------------------------|----------------|--------|
| Main Method               | 0.7593         | 0.0776 |
| Main Method (Size Filter) | 0.7670         | 0.0767 |
| Supervised                | 0.8907         | 0.0298 |

Table 4.7: Post-training performance comparison for main method and supervised baseline. Supervised: Training with the actual DUTSTR ground truths.

|                              | $max\_F_\beta$ | MAE    |
|------------------------------|----------------|--------|
| Main Method                  | 0.7593         | 0.0776 |
| Main Method (Size Filter)    | 0.7670         | 0.0767 |
| No Pretraining               | 0.5519         | 0.2082 |
| No Pretraining (Size Filter) | 0.5526         | 0.2111 |

Table 4.8: Post-training performance comparison for main method and no pretraining baseline. No Pretraining: Training on the main method without a pretrained PVTv2 backbone.

|                                   | $max\_F_\beta$ | MAE    |
|-----------------------------------|----------------|--------|
| Main Method                       | 0.7593         | 0.0776 |
| Main Method (Size Filter)         | 0.7670         | 0.0767 |
| Eigenvectors                      | 0.7070         | 0.1130 |
| Eigenvectors (Spatially Adjusted) | 0.7124         | 0.1216 |

Table 4.9: Post-training performance comparison for main method and eigenvectors.

## 4.6 Ablations

For loss function, void pixels, local K, unreliable cluster filtering, CRF, and corner-based center prior ablations see Tables 4.10, 4.11, 4.12, 4.14, 4.15 and 4.16 respectively. Since there is not much variation in the performance when using BCE or CCE compared to using the default loss function, we choose to continue using the original loss for all other experiments. One interesting result is that compared to the main method, not using unreliable cluster filtering is better on the pseudo-ground truth but the post-training predictions perform worse. While using void pixels does not seem to affect performance much, using unreliable cluster filtering and a CRF can help significantly.

| Ablation                     | $max\_F_\beta$ | MAE    |
|------------------------------|----------------|--------|
| Original Loss                | 0.7593         | 0.0776 |
| Original Loss (Size Filter)  | 0.7670         | 0.0767 |
| BCE                          | 0.7642         | 0.0819 |
| BCE (Size Filter)            | 0.7696         | 0.0809 |
| CCE {0.2, 0.8}               | 0.7616         | 0.0754 |
| CCE {0.2, 0.8} (Size Filter) | 0.7697         | 0.0741 |

Table 4.10: Loss function comparison. The original loss and BCE were used with void pixels and 20% random background labels while CCE was used with the indicated soft labels.

## Results on Test Data without SelfReformer Training

We also evaluate the main method and eigenvectors on the full DUTSTE dataset directly, without the final step of training a SelfReformer. Note that in this case, we do not remove any images from the test dataset if they contain no salient object. The results are in 4.17.

|                           |                |        |
|---------------------------|----------------|--------|
| Ablation                  | $max\_F_\beta$ | MAE    |
| Main Method               | 0.7593         | 0.0776 |
| Main Method (Size Filter) | 0.7670         | 0.0767 |
| No Void                   | 0.7561         | 0.0807 |
| No Void (Size Filter)     | 0.7624         | 0.0796 |

Table 4.11: Void pixel ablation.

|                           |                |        |
|---------------------------|----------------|--------|
| Ablation                  | $max\_F_\beta$ | MAE    |
| Main Method               | 0.7593         | 0.0776 |
| Main Method (Size Filter) | 0.7670         | 0.0767 |
| K = 2                     | 0.7402         | 0.098  |
| K = 2 (Size Filter)       | 0.7480         | 0.0986 |

Table 4.12: Local K ablation.

|                           |                |        |
|---------------------------|----------------|--------|
| Ablation                  | $max\_F_\beta$ | MAE    |
| Main Method               | 0.7593         | 0.0776 |
| Main Method (Size Filter) | 0.7670         | 0.0767 |
| K = 10                    | 0.7668         | 0.0742 |
| K = 10 (Size Filter)      | 0.7684         | 0.0742 |
| K = 20                    | 0.7643         | 0.0759 |
| K = 20 (Size Filter)      | 0.7666         | 0.0757 |

Table 4.13: Global K ablation.

|                                               |                |        |
|-----------------------------------------------|----------------|--------|
| Ablation                                      | $max\_F_\beta$ | MAE    |
| Main Method                                   | 0.7593         | 0.0776 |
| Main Method (Size Filter)                     | 0.7670         | 0.0767 |
| No Unreliable Cluster Filtering               | 0.7298         | 0.0905 |
| No Unreliable Cluster Filtering (Size Filter) | 0.7355         | 0.0901 |

Table 4.14: Unreliable Cluster Filtering ablation.

The performance decreases significantly compared to DUTSTR, because DUTSTE dataset is harder and now there is no filtering of images without the salient object. These results also show that training SelfReformer on pseudo-ground truth is crucial for obtaining good results.

|             |                |        |
|-------------|----------------|--------|
| Ablation    | $max\_F_\beta$ | MAE    |
| Main Method | 0.7593         | 0.0776 |
| No CRF      | 0.6822         | 0.1011 |

Table 4.15: CRF ablation. Note - The 2% size filter did not change the ground truth when not using a CRF.

|                           |                |        |
|---------------------------|----------------|--------|
| Ablation                  | $max\_F_\beta$ | MAE    |
| Main Method               | 0.7593         | 0.0776 |
| Main Method (Size Filter) | 0.7670         | 0.0767 |
| Corner Cut                | 0.6671         | 0.1343 |
| Corner Cut (Size Filter)  | 0.6728         | 0.1383 |

Table 4.16: Corner-based center prior ablation.

|                                   |                |        |
|-----------------------------------|----------------|--------|
| Method                            | $max\_F_\beta$ | MAE    |
| Main Method                       | 0.4160         | 0.1099 |
| Eigenvectors                      | 0.6477         | 0.1292 |
| Eigenvectors (Spatially Adjusted) | 0.6528         | 0.1432 |

Table 4.17: DUTSTE Baseline. The performance of the main method, default eigenvectors and spatially adjusted eigenvectors.

## 4.7 Qualitative Results

See Figure 4.1 for some examples of the predictions made by the SelfReformer trained on our pseudo-ground truth.

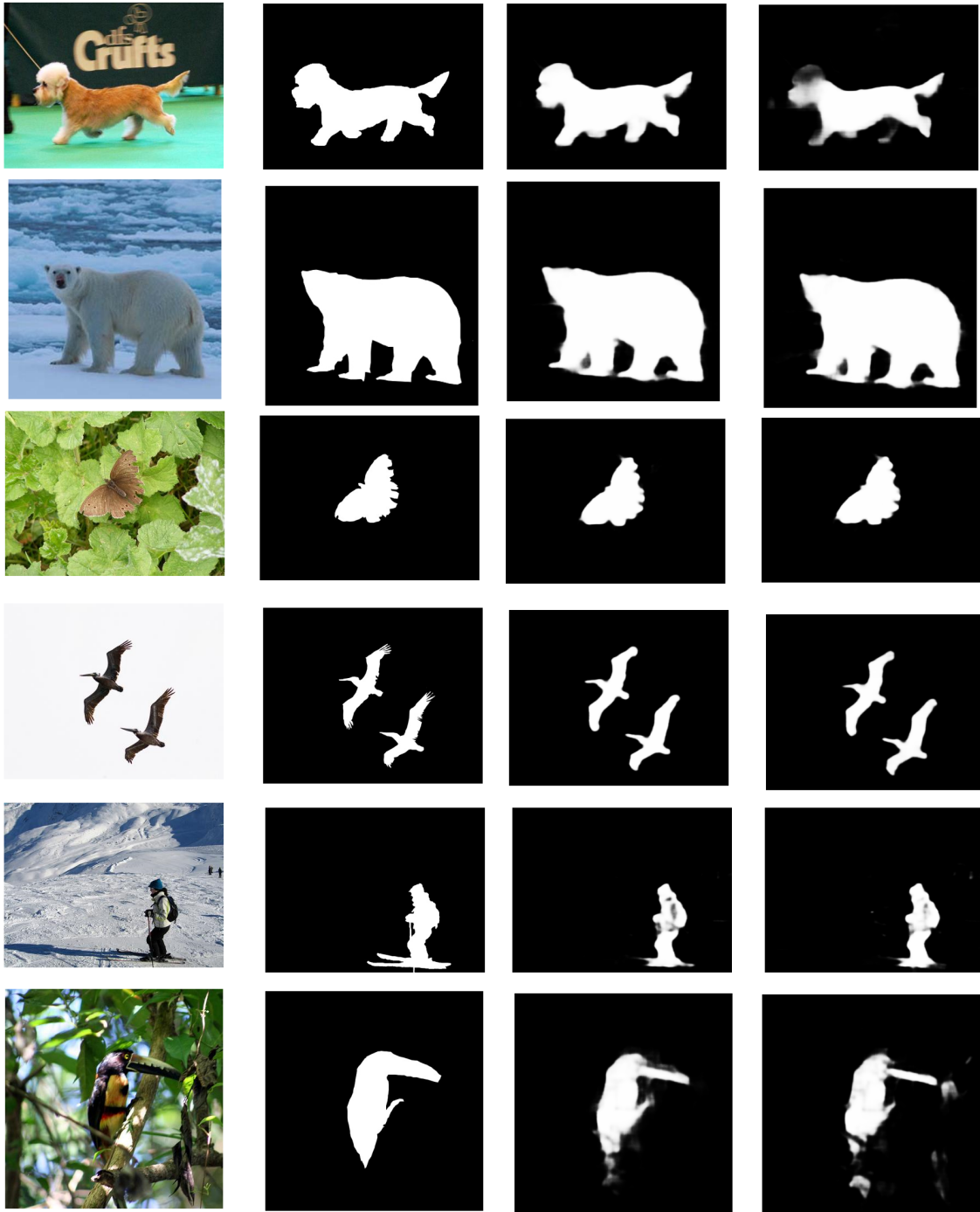


Figure 4.1: From left to right - Input Image, Ground Truth, Prediction by Main Method, Prediction by Local Spatial Centeredness

# Chapter 5

## Conclusion

In this thesis we explored the problem of unsupervised Salient Object Detection (SOD). Unsupervised feature learning in particular is a good starting point for various computer vision tasks which lead us to consider the prospect of using DINO features that have shown to be particularly strong salient representations. While ViT and DINO were originally trained to do image classification, the features produced by DINO are self-supervised which can be considered more pure representations than the ones produced by a model that has been influenced by the biases of supervision.

While Melas-Kyriazi *et al.* [80] and TokenCut [120] explore how to use spectral clustering on these features to do various computer vision tasks, in this thesis we strived to cover a more comprehensive investigation of the relationship between clustering and DINO features. This was done through our experiments with a multitude of methods for using DINO features where we attempted to find the clusters that are rich in salient information. Even with one of the more simple clustering algorithms, K-means, we were able to obtain promising results that could indicate that the features themselves are directly clusterable or have extremely good clustering properties.

Future work could try to continue down the global clustering methods path for further improvement. Another path could be to ablate the effectiveness of other clustering algorithms such as Gaussian Mixture Models (GMMs). Finally, the ultimate question for future work could be to go all the way back to the beginning and ask why DINO features are particularly useful. To suggest something that might be a more feasible version of this question, why do the “key” tokens from DINO work the best?

# References

- [1] 3B1B. <https://www.youtube.com/watch?v=eMlx5fFNoYc>.
- [2] Rameen Abdal, Peihao Zhu, Niloy Mitra, and Peter Wonka. Labels4Free: Unsupervised segmentation using StyleGAN. In *ICCV*, 2021.
- [3] Radhakrishna Achanta, Sheila Hemami, Francisco Estrada, and Sabine Susstrunk. Frequency-tuned salient region detection. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1597–1604, 2009.
- [4] Haitham Afan, Ahmedbahaaaldin Ibrahim Ahmed Osman, Yusuf Essam, Al-Mahfoodh Ali Najah Ahmed, Yuk Huang, Ozgur Kisi, Mohsen Sherif, Ahmed Sefelnasr, Kwok Chau, and Ahmed El-Shafie. Modeling the fluctuations of groundwater level by employing ensemble deep learning techniques. *Engineering Applications of Computational Fluid Mechanics*, 15:1420–1439, 09 2021.
- [5] Relja Arandjelović and Andrew Zisserman. Object discovery with a copy-pasting GAN. [Online] arXiv 2019.
- [6] Sara Atito, Muhammad Awais, and Josef Kittler. SiT: Self-supervised vision transformer, [Online] arXiv 2022.
- [7] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEiT: BERT pre-training of image transformers, [Online] arXiv 2022.
- [8] Jonathan T. Barron and Ben Poole. The fast bilateral solver. [Online] arXiv 2016.
- [9] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020.

- [10] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS*, 2020.
- [11] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.
- [12] Jeff Cheeger. *A Lower Bound for the Smallest Eigenvalue of the Laplacian*, pages 195–200. Princeton University Press, Princeton, 1971.
- [13] Chun-Fu Chen, Quanfu Fan, and Rameswar Panda. CrossViT: Cross-attention multi-scale vision transformer for image classification. In *ICCV*, 2021.
- [14] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2018.
- [15] Mickaël Chen, Thierry Artières, and Ludovic Denoyer. Unsupervised object segmentation by redrawing. In *NeurIPS*, 2019.
- [16] Tianshui Chen, Liang Lin, Lingbo Liu, Xiaonan Luo, and Xuelong Li. DISC: Deep image saliency computing via progressive representation learning. *IEEE Transactions on Neural Networks and Learning Systems*, 27(6):1135–1149, 2016.
- [17] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.
- [18] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning, [Online] arXiv 2020.
- [19] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *CVPR*, 2021.
- [20] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *ICCV*, 2021.
- [21] Ming-Ming Cheng, Niloy J. Mitra, Xiaolei Huang, Philip H. S. Torr, and Shi-Min Hu. Global contrast based salient region detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):569–582, 2015.

- [22] Minsu Cho, Suha Kwak, Cordelia Schmid, and Jean Ponce. Unsupervised object discovery and localization in the wild: Part-based matching with bottom-up region proposals. In *CVPR*, 2015.
- [23] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. In *NeurIPS*, 2021.
- [24] Edo Collins, Radhakrishna Achanta, and Sabine Ssstrunk. Deep feature factorization for concept discovery. In *ECCV*, 2018.
- [25] Zhigang Dai, Bolun Cai, Yugeng Lin, and Junying Chen. Unsupervised pre-training for detection transformers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 1–11, 2022.
- [26] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.
- [27] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.
- [28] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015.
- [29] W. E. Donath and A. J. Hoffman. Lower bounds for the partitioning of graphs. *IBM Journal of Research and Development*, 17(5):420–425, 1973.
- [30] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.
- [31] Linus Ericsson, Henry Gouk, Chen Change Loy, and Timothy M. Hospedales. Self-supervised representation learning: Introduction, advances, and challenges. *IEEE Signal Processing Magazine*, 39(3):42–62, 2022.
- [32] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

- [33] Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C. Platt, C. Lawrence Zitnick, and Geoffrey Zweig. From captions to visual concepts and back. In *CVPR*, 2015.
- [34] Pedro Felzenszwalb and Daniel Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59:167–181, 09 2004.
- [35] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(2):298–305, 1973.
- [36] Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, and Luc Van Gool. Unsupervised semantic segmentation by contrasting object mask proposals. In *ICCV*, 2021.
- [37] Yue Gao, Meng Wang, Zheng-Jun Zha, Jialie Shen, Xuelong Li, and Xindong Wu. Visual-textual joint relevance learning for tag-based social image search. *IEEE Transactions on Image Processing*, 22(1):363–376, 2013.
- [38] Andreas Geiger, P Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: the KITTI dataset. *The International Journal of Robotics Research*, 32:1231–1237, 09 2013.
- [39] Spyros Gidaris, Andrei Bursuc, Nikos Komodakis, Patrick Pérez, and Matthieu Cord. Learning representations by predicting bags of visual words. In *CVPR*, 2020.
- [40] Spyros Gidaris, Andrei Bursuc, Gilles Puy, Nikos Komodakis, Matthieu Cord, and Patrick Pérez. OBoW: Online bag-of-visual-words generation for self-supervised learning. In *CPVR*, 2021.
- [41] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018.
- [42] Ben Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. LeViT: a vision transformer in ConvNet’s clothing for faster inference. In *ICCV*, 2021.
- [43] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. In *NIPS*, 2020.

- [44] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In Yee Whye Teh and Mike Titterton, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 297–304, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.
- [45] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022.
- [46] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.
- [47] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.
- [48] Olivier J. Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, S. M. Ali Eslami, and Aaron van den Oord. Data-efficient image recognition with contrastive predictive coding. In *ICML*, 2020.
- [49] Byeongho Heo, Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Junsuk Choe, and Seong Joon Oh. Rethinking spatial dimensions of vision transformers. In *ICCV*, 2021.
- [50] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NIPS*, 2014.
- [51] Kuang-Jui Hsu, Yen-Yu Lin, and Yung-Yu Chuang. Co-attention CNNs for unsupervised object co-segmentation. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 748–756. International Joint Conferences on Artificial Intelligence Organization, 7 2018.
- [52] Zilong Huang, Xinggang Wang, Yunchao Wei, Lichao Huang, Humphrey Shi, Wenyu Liu, and Thomas S. Huang. CCNet: Criss-cross attention for semantic segmentation. In *ICCV*, 2019.
- [53] Jyh-Jing Hwang, Stella X. Yu, Jianbo Shi, Maxwell D. Collins, Tien-Ju Yang, Xiao Zhang, and Liang-Chieh Chen. SegSort: Segmentation by discriminative sorting of segments. In *ICCV*, 2019.

- [54] Xu Ji, João F. Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *ICCV*, 2019.
- [55] Huaizu Jiang, Jingdong Wang, Zejian Yuan, Yang Wu, Nanning Zheng, and Shipeng Li. Salient object detection: A discriminative regional feature integration approach. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2083–2090, 2013.
- [56] Armand Joulin, Francis Bach, and Jean Ponce. Discriminative clustering for image co-segmentation. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1943–1950, 2010.
- [57] Armand Joulin, Francis Bach, and Jean Ponce. Multi-class cosegmentation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 542–549, 2012.
- [58] Yannis Kalantidis, Mert Bulent Sariyildiz, Noe Pion, Philippe Weinzaepfel, and Diane Larlus. Hard negative mixing for contrastive learning. In *NeurIPS*, 2020.
- [59] Jiwhan Kim, Dongyoon Han, Yu-Wing Tai, and Junmo Kim. Salient region detection via high-dimensional color transform. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 883–890, 2014.
- [60] Yuqiu Kong, Lijun Wang, Xiuping Liu, Huchuan Lu, and Xiang Ruan. Pattern mining saliency. In *ECCV 2016*, volume 9910, pages 583–598, 10 2016.
- [61] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.
- [62] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected CRFs with Gaussian edge potentials. In *NIPS*, 2012.
- [63] Jason Kuen, Zhenhua Wang, and Gang Wang. Recurrent attentional networks for saliency detection. In *CVPR*, 2016.
- [64] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A Lite BERT for self-supervised learning of language representations. In *ICLR*, 2020.
- [65] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

- [66] Yann Lecun and Y. Bengio. Convolutional networks for images, speech, and time-series. In *The Handbook of Brain Theory and Neural Networks*, 01 1995.
- [67] Gayoung Lee, Yu-Wing Tai, and Junmo Kim. Deep saliency with encoded low level distance map and high level features. In *CVPR*, 2016.
- [68] Guanbin Li and Yizhou Yu. Visual saliency based on multiscale deep features. In *CVPR*, 2015.
- [69] Xi Li, Liming Zhao, Lina Wei, Ming-Hsuan Yang, Fei Wu, Yueting Zhuang, Haibin Ling, and Jingdong Wang. DeepSaliency: Multi-task deep neural network model for salient object detection. *IEEE Transactions on Image Processing*, 25(8):3919–3930, August 2016.
- [70] Yin Li, Xiaodi Hou, Christof Koch, James M. Rehg, and Alan L. Yuille. The secrets of salient object segmentation. In *CVPR*, 2014.
- [71] Zhaowen Li, Zhiyang Chen, Fan Yang, Wei Li, Yousong Zhu, Chaoyang Zhao, Rui Deng, Liwei Wu, Rui Zhao, Ming Tang, and Jinqiao Wang. MST: Masked Self-Supervised Transformer for visual representation. In *NIPS*, 2021.
- [72] Nian Liu and Junwei Han. DHSNet: Deep hierarchical saliency network for salient object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 678–686, 2016.
- [73] Nian Liu, Ni Zhang, Kaiyuan Wan, Ling Shao, and Junwei Han. Visual Saliency Transformer. In *ICCV*, 2021.
- [74] Tie Liu, Jian Sun, Nan-Ning Zheng, Xiaoou Tang, and Heung-Yeung Shum. Learning to detect a salient object. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.
- [75] Yanbin Liu and Stephen Gould. Unsupervised dense prediction using differentiable normalized cuts. In *ECCV*, 2024.
- [76] Yen-Cheng Liu, Chih-Yao Ma, Zijian He, Chia-Wen Kuo, Kan Chen, Peizhao Zhang, Bichen Wu, Zsolt Kira, and Peter Vajda. Unbiased teacher for semi-supervised object detection. In *ICLR*, 2021.
- [77] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021.

- [78] Song Lu, Vijay Mahadevan, and Nuno Vasconcelos. Learning optimal seeds for diffusion-based salient object detection. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2790–2797, 2014.
- [79] Warren Mcculloch and Walter Pitts. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:127–147, 1943.
- [80] Luke Melas-Kyriazi, Christian Rupprecht, Iro Laina, and Andrea Vedaldi. Deep spectral methods: A surprisingly strong baseline for unsupervised semantic segmentation and localization. In *CVPR*, 2022.
- [81] M. Minsky and S. Papert. *Perceptrons; an Introduction to Computational Geometry*. MIT Press, 1969.
- [82] Muzammal Naseer, Kanchana Ranasinghe, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Intriguing properties of vision transformers. In *NIPS*, 2021.
- [83] Andrew Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001.
- [84] Duc Tam Nguyen, Maximilian Dax, Chaithanya Kumar Mummadi, Thi Phuong Nhung Ngo, Thi Hoai Phuong Nguyen, Zhongyu Lou, and Thomas Brox. DeepUSPS: Deep robust unsupervised saliency prediction with self-supervision. In *NeurIPS*, 2019.
- [85] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016.
- [86] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016.
- [87] Pedro O. Pinheiro, Amjad Almahairi, Ryan Y. Benmalek, Florian Golemo, and Aaron Courville. Unsupervised learning of dense visual representations. In *NIPS*, 2020.
- [88] Xuebin Qin, Zichen Zhang, Chenyang Huang, Masood Dehghan, Osmar R. Zaiane, and Martin Jagersand. U2-Net: Going deeper with nested U-structure for salient object detection. *Pattern Recognition*, 106:107404, 2020.
- [89] QuantStart. <https://www.quantstart.com/articles/introduction-to-artificial-neural-networks-and-the-perceptron/>.

- [90] Zhongzheng Ren, Zhiding Yu, Xiaodong Yang, Ming-Yu Liu, Yong Jae Lee, Alexander G. Schwing, and Jan Kautz. Instance-aware, context-focused, and memory-efficient weakly supervised object detection. In *CVPR*, 2020.
- [91] F. Rosenblatt. *The Perceptron, a Perceiving and Recognizing Automaton Project Para.* Report: Cornell Aeronautical Laboratory. Cornell Aeronautical Laboratory, 1957.
- [92] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.
- [93] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.
- [94] Jianping Shi, Qiong Yan, Li Xu, and Jiaya Jia. Hierarchical image saliency detection on Extended CSSD. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(4):717–729, 2016.
- [95] Zhenmei Shi, Junyi Wei, and Yingyu Liang. A theoretical analysis on feature learning in neural networks: Emergence from inputs and advantage over fixed features. In *ICLR*, 2022.
- [96] Jamie Shotton, Matthew Johnson, and Roberto Cipolla. Semantic texton forests for image categorization and segmentation. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.
- [97] Oriane Siméoni, Gilles Puy, Huy V. Vo, Simon Roburin, Spyros Gidaris, Andrei Bursuc, Patrick Pérez, Renaud Marlet, and Jean Ponce. Localizing objects with self-supervised transformers and no labels. In *BMVC*, 2021.
- [98] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. In *ICCV*, 2021.
- [99] Richard Szeliski. *Computer Vision*. Springer London, 2010.
- [100] Kevin Tang, Armand Joulin, Li-Jia Li, and Li Fei-Fei. Co-localization in real-world images. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1464–1471, 2014.
- [101] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *ECCV*, 2020.

- [102] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021.
- [103] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *ICCV*, 2021.
- [104] Anne M. Treisman and Garry Gelade. A feature-integration theory of attention. *Cognitive Psychology*, 12(1):97–136, 1980.
- [105] Shikhar Tuli, Ishita Dasgupta, Erin Grant, and Thomas L. Griffiths. Are convolutional neural networks or transformers more like human vision? In *CogSci*, 2021.
- [106] Stanford University. <https://www.youtube.com/watch?v=bNb2fEVKeEo>.
- [107] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, [Online] arXiv 2019.
- [108] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.
- [109] Sara Vicente, Carsten Rother, and Vladimir Kolmogorov. Object cosegmentation. In *CVPR 2011*, pages 2217–2224, 2011.
- [110] Huy V. Vo, Francis Bach, Minsu Cho, Kai Han, Yann LeCun, Patrick Perez, and Jean Ponce. Unsupervised image matching and object discovery as optimization. In *CVPR*, 2019.
- [111] Huy V. Vo, Patrick Pérez, and Jean Ponce. Toward unsupervised, multi-object discovery in large-scale image collections. In *ECCV*, 2020.
- [112] Huy V. Vo, Elena Sizikova, Cordelia Schmid, Patrick Pérez, and Jean Ponce. Large-scale unsupervised object discovery. In *NeurIPS*, 2021.
- [113] Andrey Voynov, Stanislav Morozov, and Artem Babenko. Object segmentation without labels with large-scale generative models. In *ICML*, 2021.
- [114] Lijun Wang, Huchuan Lu, Xiang Ruan, and Ming-Hsuan Yang. Deep networks for saliency detection via local estimation and global search. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3183–3192, 2015.

- [115] Lijun Wang, Huchuan Lu, Yifan Wang, Mengyang Feng, Dong Wang, Baocai Yin, and Xiang Ruan. Learning to detect salient objects with image-level supervision. In *CVPR*, 2017.
- [116] Linzhao Wang, Lijun Wang, Huchuan Lu, Pingping Zhang, and Xiang Ruan. Salient object detection with recurrent fully convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(7):1734–1746, 2019.
- [117] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid Vision Transformer: A versatile backbone for dense prediction without convolutions. In *ICCV*, 2021.
- [118] Xiang Wang, Shaodi You, Xi Li, and Huimin Ma. Weakly-supervised semantic segmentation by iteratively mining common object features. In *CVPR*, 2018.
- [119] Xinlong Wang, Rufeng Zhang, Chunhua Shen, Tao Kong, and Lei Li. Dense contrastive learning for self-supervised visual pre-training. In *CVPR*, 2021.
- [120] Yangtao Wang, Xi Shen, Shell Hu, Yuan Yuan, James Crowley, and Dominique Vaufreydaz. Self-supervised transformers for unsupervised object discovery using normalized cut. In *CVPR*, 2022.
- [121] Jun Wei, Shuhui Wang, and Qingming Huang. F3Net: Fusion, feedback and focus for salient object detection. In *AAAI*, 2020.
- [122] Xiu-Shen Wei, Chen-Lin Zhang, Jianxin Wu, Chunhua Shen, and Zhi-Hua Zhou. Unsupervised object discovery and co-localization by deep descriptor transforming. In *IJCAI*, 2017.
- [123] Yichen Wei, Fang Wen, Wangjiang Zhu, and Jian Sun. Geodesic saliency using background priors. In *Proceedings of the 12th European conference on Computer Vision - Volume Part III*, volume 7574, pages 29–42, 10 2012.
- [124] Yunchao Wei, Jiashi Feng, Xiaodan Liang, Ming-Ming Cheng, Yao Zhao, and Shuicheng Yan. Object region mining with adversarial erasing: A simple classification to semantic segmentation approach. In *CVPR*, 2017.
- [125] Bichen Wu, Alvin Wan, Forrest Iandola, Peter H. Jin, and Kurt Keutzer. SqueezeDet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. In *CVPR*, 2017.

- [126] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. CvT: Introducing convolutions to vision transformers. In *ICCV*, 2021.
- [127] Zhirong Wu, Yuanjun Xiong, Stella Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance-level discrimination. In *CVPR*, 2018.
- [128] Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3485–3492, 2010.
- [129] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M. Alvarez, and Ping Luo. SegFormer: Simple and efficient design for semantic segmentation with transformers. In *NeurIPS*, 2021.
- [130] Zhenda Xie, Yutong Lin, Zhuliang Yao, Zheng Zhang, Qi Dai, Yue Cao, and Han Hu. Self-supervised learning with swin transformers. In *CVPR*, 2021.
- [131] Zhenda Xie, Yutong Lin, Zheng Zhang, Yue Cao, Stephen Lin, and Han Hu. Propagate yourself: Exploring pixel-level consistency for unsupervised visual representation learning. In *CVPR*, 2021.
- [132] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015.
- [133] Qiong Yan, Li Xu, Jianping Shi, and Jiaya Jia. Hierarchical saliency detection. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1155–1162, 2013.
- [134] Chuan Yang, Lihe Zhang, Huchuan Lu, Xiang Ruan, and Ming-Hsuan Yang. Saliency detection via graph-based manifold ranking. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3166–3173, 2013.
- [135] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zihang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-Token ViT: Training vision transformers from scratch on ImageNet. In *ICCV*, 2021.
- [136] Yi Ke Yun and Weisi Lin. Towards a complete and detail-preserved salient object detection. *IEEE Transactions on Multimedia*, pages 1–15, 2023.

- [137] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow Twins: Self-supervised learning via redundancy reduction. In *ICML*, 2021.
- [138] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *CVPR*, 2022.
- [139] Dingwen Zhang, Deyu Meng, Long Zhao, and Junwei Han. Bridging saliency detection to weakly supervised object detection based on self-paced curriculum learning. In *IJCAI*, 2016.
- [140] Jianming Zhang, Stan Sclaroff, Zhe Lin, Xiaohui Shen, Brian Price, and Radomir Mech. Minimum barrier salient object detection at 80 fps. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.
- [141] Jing Zhang, Tong Zhang, Yuchao Dai, Mehrtash Harandi, and Richard Hartley. Deep unsupervised saliency detection: A multiple noisy labeling perspective. In *CVPR*, 2018.
- [142] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization. In *ECCV*, 2016.
- [143] Richard Zhang, Phillip Isola, and Alexei A. Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *CVPR*, 2017.
- [144] Runsheng Zhang, Yaping Huang, Mengyang Pu, Jian Zhang, Qingji Guan, Qi Zou, and Haibin Ling. Object discovery from a single unlabeled image by mining frequent itemsets with multi-scale features. *IEEE Transactions on Image Processing*, 29:8606–8621, 2020.
- [145] Xiao Zhang and Michael Maire. Self-supervised visual representation learning from hierarchical grouping. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 16579–16590. Curran Associates, Inc., 2020.
- [146] Zhongwen Zhang and Yuri Boykov. Collision cross-entropy for soft class labels and deep clustering. [Online] arXiv 2023.
- [147] Rui Zhao, Wanli Ouyang, Hongsheng Li, and Xiaogang Wang. Saliency detection by multi-context deep learning. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1265–1274, 2015.

- [148] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip H. S. Torr, and Li Zhang. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *CVPR*, 2021.
- [149] Daquan Zhou, Bingyi Kang, Xiaojie Jin, Linjie Yang, Xiaochen Lian, Zihang Jiang, Qibin Hou, and Jiashi Feng. DeepViT: Towards deeper vision transformer, [Online] arXiv 2021.
- [150] Heqin Zhu, Xu Sun, Yuexiang Li, Kai Ma, S. Kevin Zhou, and Yefeng Zheng. DFTR: Depth-supervised fusion transformer for salient object detection. [Online] arXiv 2022.
- [151] Wangjiang Zhu, Shuang Liang, Yichen Wei, and Jian Sun. Saliency optimization from robust background detection. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2814–2821, 2014.
- [152] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: Deformable transformers for end-to-end object detection. In *ICLR*, 2021.
- [153] Wenbin Zou and Nikos Komodakis. HARF: Hierarchy-associated rich features for salient object detection. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 406–414, 2015.