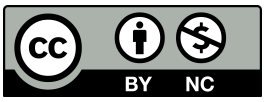
Unsupervised Object Segmentation with Generative Models

Inauguraldissertation
der Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

vorgelegt von
Adam Jakub Bielski
von Polen

Leiter der Arbeit:
Prof. Dr. Paolo Favaro
Institut für Informatik



Unsupervised Object Segmentation with Generative Models

Inauguraldissertation
der Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

vorgelegt von

Adam Jakub Bielski
von Polen

Leiter der Arbeit:
Prof. Dr. Paolo Favaro
Institut für Informatik

Von der Philosophisch-naturwissenschaftlichen Fakultät angenommen.

Bern, 24.04.2024

Der Dekan:
Prof. Dr. M. Herwegh

Abstract

Advances in computer vision have transformed how we interact with technology, driven by significant breakthroughs in scalable deep learning and the availability of large datasets. These technologies now play a crucial role in various applications, from improving user experience through applications like organizing digital photo libraries, to advancing medical diagnostics and treatments. Despite these valuable applications, the creation of annotated datasets remains a significant bottleneck. It is not only costly and labor-intensive but also prone to inaccuracies and human biases. Moreover, it often requires specialized knowledge or careful handling of sensitive information. Among the tasks in computer vision, image segmentation particularly highlights these challenges, with its need for precise pixel-level annotations. This context underscores the need for unsupervised approaches in computer vision, which can leverage the large volumes of unlabeled images produced every day.

This thesis introduces several novel methods for learning fully unsupervised object segmentation models using only collections of images. Unlike much prior work, our approaches are effective on complex real-world images and do not rely on any form of annotations, including pre-trained supervised networks, bounding boxes, or class labels. We identify and leverage intrinsic properties of objects – most notably, the cohesive movement of object parts – as powerful signals for driving unsupervised object segmentation. Utilizing innovative generative adversarial models, we employ this principle to either generate segmented objects or directly segment them in a manner that allows for realistic movement within scenes. Our work demonstrates how such generated data can train a segmentation model that effectively generalizes to real-world images. Furthermore, we introduce a method that, in conjunction with recent advances in self-supervised learning, achieves state-of-the-art results in unsupervised object segmentation. Our methods rely on the effectiveness of Generative Adversarial Networks, which are known to be challenging to train and exhibit mode collapse. We propose a new, more principled GAN loss, whose gradients encourage the generator model to explore missing modes in its distribution, addressing these limitations and enhancing the robustness of generative models.

Acknowledgments

First and foremost, I would like to thank my PhD advisor, Prof. Dr. Paolo Favaro. Our paths crossed at the poster session at the CVPR conference in Salt Lake City in 2018, at a time when I was searching of a PhD program. At the same conference, I had the opportunity to listen to him as a panelist at the Beyond Supervised Learning workshop and was deeply inspired by his perspective on machine learning. I am profoundly grateful for the opportunity he has given me to pursue my passion in machine learning research. His support, guidance, and the countless hours we have spent brainstorming and discussing ideas have been invaluable. His determination to see ideas through to success is something I often remind myself of whenever I face doubts.

A special thanks goes to Prof. Dr. Amir Zamir and Prof. Dr. Torsten Braun for serving as thesis examiners. I appreciate their valuable feedback.

I want to acknowledge all the members and friends of Computer Vision Group at the University of Bern that I had the pleasure to meet and work with, including Adrian Wälchli, Abdelhak Lemkhenter, Simon Jenni, Givi Meishvili, Sepehr Sameni, Llukman Çerkezi, Aram Davtyan, Alp Eren Sari, Viktor Shipitsin, Hamadi Chihaoui, Josué Page Vizcaíno, Tomoki Watanabe, Antonino Furnari, Qiyang Hu, Xiaochen Wang, Renato Maria Prisco, Luigi Fiorillo, Athanasios Charisoudis, Florence Aellen, and Dragana Esser. The laughter and insights shared during our coffee breaks and lunches added so much value to my PhD experience.

My journey to completing this PhD was made possible by the support and encouragement from my friends, Monika A., Wojciech S., Anna M., Maria W., Agnieszka K., Berenika K., Robin W., Mateusz G., Matuesz N., Filip K. and Daniel C. Thank you for being there during the toughest times.

A final word of thanks to my beloved family: my parents, Krystyna and Janusz, and my siblings, Anna and Daniel. My mom, with her passion for math and teaching, has been my inspiration, showing me the joy of learning and the importance of education from the start.

Contents

1 Introduction **15**
- 1.1 Limitations of Supervised Object Segmentation . . . 17
- 1.2 Human Perception of Objects . . . 17
- 1.3 Leveraging generative models . . . 19
- 1.4 Thesis contributions . . . 20
  - 1.4.1 Chapter outline . . . 20

2 Background **23**
- 2.1 Image Segmentation Formulation . . . 23
- 2.2 Neural Networks for Image Segmentation . . . 24
  - 2.2.1 Convolutional Neural Networks (CNNs) . . . 24
  - 2.2.2 Encoder-Decoder Architectures . . . 24
  - 2.2.3 Instance Segmentation with Mask R-CNN . . . 25
  - 2.2.4 Attention Mechanisms and Transformer Models . . . 25
- 2.3 Unsupervised Object Segmentation Methods . . . 26
  - 2.3.1 Clustering and Hand-crafted Methods . . . 26
  - 2.3.2 Mutual Information and Scene Decomposition . . . 26
  - 2.3.3 Generative Methods . . . 27
  - 2.3.4 Self-Supervised Pre-trained Models . . . 28
- 2.4 Generative Adversarial Networks . . . 29
  - 2.4.1 Problem Formulation . . . 29
  - 2.4.2 Limitations and Advancements . . . 30

3 Emergence of Object Segmentation in Perturbed Generative Models **33**
- 3.1 Related Work . . . 35
- 3.2 Learning to Segment without Supervision . . . 36
  - 3.2.1 A Generative Model of Layered Scenes . . . 37
  - 3.2.2 Learning through Model Perturbation . . . 37
  - 3.2.3 Object Segmentation via Autoencoding Constraints . . . 38
- 3.3 Implementation . . . 39

3.4 Experiments . . . 40
3.4.1 Ablation study . . . 42
3.4.2 Segmenting real images . . . 43
3.5 Discussion . . . 45

4 Unsupervised Learning of Object Segmentation From Perturbed Generative Models 47
4.1 Learning to Segment without Supervision . . . 49
4.1.1 A Generative Model of Layered Scenes through Model Perturbation 49
4.1.2 Object Segmentation Trained on Generated Data . . . 50
4.2 Implementation . . . 52
4.3 Experiments . . . 53
4.3.1 Datasets . . . 53
4.3.2 Ablation study . . . 53
4.3.3 Segmentation results . . . 54
4.4 Discussion . . . 59

5 MOVE: Unsupervised Movable Object Segmentation and Detection 61
5.1 Background . . . 63
5.2 Method . . . 63
5.2.1 Segmenter . . . 66
5.2.2 Differentiable inpainting . . . 66
5.2.3 Adversarial training . . . 69
5.3 Implementation . . . 70
5.4 Experiments . . . 70
5.4.1 Unsupervised saliency segmentation . . . 70
5.4.2 Single-object discovery . . . 73
5.4.3 Ablation study . . . 75
5.5 Discussion . . . 77

6 Generative Adversarial Learning via Kernel Density Discrimination 85
6.1 Background . . . 87
6.2 Kernel Density Discrimination . . . 87
6.2.1 Improving KDE through Data Augmentation . . . 90
6.2.2 Loss Analysis . . . 90
6.2.3 Class-Conditioning Extension . . . 92
6.2.4 Regularization of the Feature Mapping . . . 92
6.2.5 KDD GAN Formulation . . . 93
6.3 Implementation . . . 93
6.4 Experiments . . . 94

6.4.1 Ablation Results 95
6.4.2 Generative Learning on CIFAR10 96
6.4.3 Generative Learning on ImageNet 97
6.5 Examples of Generated Images 99
6.6 Discussion 99

7 Conclusions **107**

Bibliography **109**

List of Figures

1.1 Examples of different image segmentation tasks 16
1.2 Object perception in infants 18

2.1 U-Net architecture 25
2.2 Masked Autoencoder 28
2.3 DINO segmentation examples 29

3.1 Learning scheme for the generative layered image representation . . . 34
3.2 Trivial solutions for the layered image representation 35
3.3 Qualitative results for the layered image generation 39
3.4 Qualitative results for the ablation study 41
3.5 Object generation on a multiple object dataset 43
3.6 Qualitative results for segmenting real images 44

4.1 Scheme for joint training of the generative model and the segmenter . 48
4.2 Qualitative results of the layered image generation on diverse datasets 55
4.3 Qualitative results of segmentation on diverse datasets 56

5.1 Exploiting inpainting and movability for unsupervised object segmentation 62
5.2 Synthetic and real images used for the adversarial training 64
5.3 Scheme for predicting and using the segmentation and inpainting masks 65
5.4 Visual comparison of the default and modified MAE inpainting 68
5.5 Qualitative results of MOVE on saliency detection datasets 72
5.6 Illustration of bilateral solver disadvantages 72
5.7 Qualitative results of object detection of MOVE 75
5.8 Sample segmentation results on ECSSD. 79
5.9 Sample segmentation results on DUTS-TE. 80
5.10 Sample segmentation results on DUT-OMRON. 81
5.11 Sample detection results on VOC07. 82
5.12 Sample detection results on VOC12. 83
5.13 Sample detection results on COCO20k. 84

6.1 Illustration of the difference between the hinge loss and KDD loss . . . 86
6.2 Sample images generated using KDD GAN on ImageNet 64×64 . . . 94
6.3 Qualitative results on CIFAR10 . . . 100
6.4 Qualitative results on Tiny ImageNet with the Hinge loss . . . 101
6.5 Qualitative results on Tiny ImageNet with the KDD loss . . . 102
6.6 Qualitative results on ImageNet 64×64 with the Hinge loss . . . 103
6.7 Qualitative results on ImageNet 64×64 with the Hinge loss . . . 104
6.8 Qualitative results on ImageNet 64×64 with the KDD loss . . . 105
6.9 Qualitative results on ImageNet 64×64 with the KDD loss . . . 106

List of Tables

3.1 Ablation study for the layered generative model 40
3.2 FID score comparison 43
3.3 Quantitative results for segmenting real images 44

4.1 Comparison of different segmenter training methods 52
4.2 Unsupervised segmentation results on the CUB-200-2011 dataset . . . 58
4.3 Unsupervised segmentation results on the Flowers102 dataset 58
4.4 Unsupervised segmentation results on the LSUN Car dataset 58
4.5 Unsupervised segmentation results on the DUTS dataset 59

5.1 Inpainting error for different MAE modes 67
5.2 Quantitative results of unsupervised salient object segmentation . . . 71
5.3 Quantitative unsupervised segmentation results on the CUB-200-2011 dataset 73
5.4 Quantitative results of MOVE for unsupervised single object discovery 74
5.5 Quantitative results for unsupervised class-agnostic object detection . 75
5.6 Ablation study for MOVE 76

6.1 Ablation study of loss parameters for KDD GAN 95
6.2 Ablation study for kernel choice and feature dimension 96
6.3 Quantitative results of KDD GAN on CIFAR10 97
6.4 Quantitative results of KDD GAN on Tiny ImageNet 98
6.5 Quantiative results of KDD GAN on ImageNet 64×64 99

Chapter 1

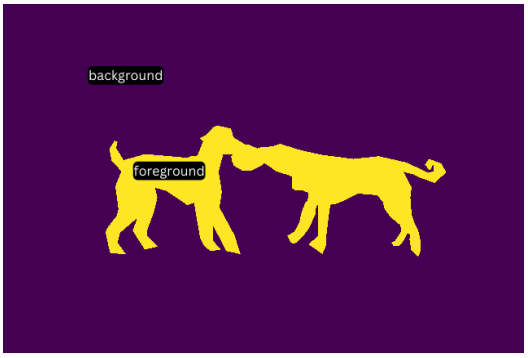
Introduction

Determining exact object boundaries through image segmentation is essential for understanding images and their context, both for humans and computer vision systems. For humans, this ability is fundamental to recognizing objects, determining how to interact with them, and anticipating the results of these interactions. In computer vision, segmentation is essential for isolating and extracting meaningful information from images by dividing them into regions that represent both different classes and individual instances of objects, as shown in Figure 1.1.

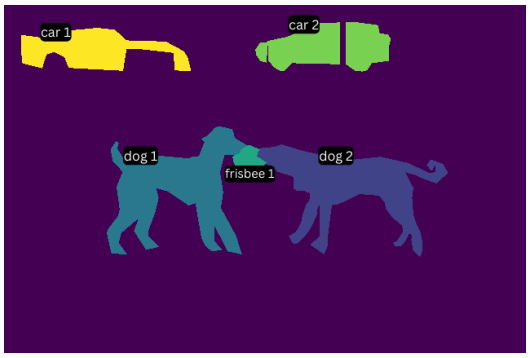
Automated image segmentation is crucial for a wide range of applications across various fields. For example, in autonomous navigation, it enables vehicles and drones to understand their environment by identifying obstacles and safe paths, enhancing navigation safety. Image manipulation benefits from segmentation through techniques such as object removal and background alteration, which are extensively used in the entertainment and advertising industries. Augmented reality (AR) applications use segmentation to merge digital objects with the real world, improving experiences in games, education, and marketing. In medical image analysis, segmentation helps in identifying and quantifying anatomical structures, aiding in diagnostics, treatment planning, and disease monitoring, which are essential for personalized healthcare. These instances are just a few examples demonstrating the critical role of image segmentation in advancing technology and improving lives.

The progress in machine learning, particularly with the advent of supervised deep learning methods powered by extensive human-annotated datasets, has greatly advanced the field of computer vision. Despite this progress, the dependence on manual annotation introduces significant challenges: it is costly, labor-intensive, and scales poorly, particularly in domains that demand specialized knowledge and strict adherence to data privacy laws, such as in medical imaging. The task of obtaining precise, pixel-level annotations for image segmentation is especially laborious, highlighting the inefficiencies of supervised methods.

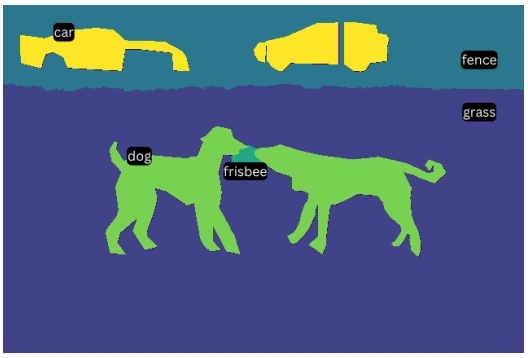
(a) Input image



(b) Foreground/salient segmentation



(c) Instance segmentation



(d) Semantic segmentation

Figure 1.1: Examples of different image segmentation tasks. Image from the COCO dataset [77].

This reliance on manual annotation raises a crucial question: Is it feasible to achieve object segmentation in a completely unsupervised manner, leveraging only the data available in collections of real images, without any form of manual annotation? This question is far from trivial, given the challenges posed by data variability, semantic ambiguity, and the crucial need for generalization across diverse scenarios.

The pursuit of unsupervised segmentation methods promises a significant leap forward in computer vision, potentially reducing the need for expensive and time-consuming annotation processes. This calls for the exploration of novel algorithms that can learn from unlabeled data, similar to how humans learn to recognize and distinguish objects without explicit instruction. Achieving this would not only streamline the process of image segmentation but also make advanced computer vision technologies more accessible and applicable across a wider range of fields.

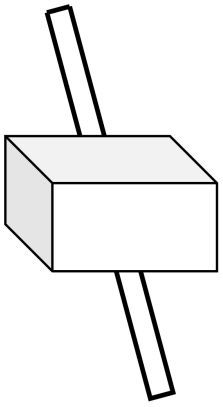
1.1 Limitations of Supervised Object Segmentation

The process of supervised object segmentation heavily depends on manually annotated data, which not only incurs high costs and demands extensive time but also requires specialized domain expertise. In the supervised setting, it is the pixel-level annotations by humans that define what constitutes an object. This underscores another fundamental limitation: this approach offers a *roundabout definition* of what an object is, filtered through human perception and interpretation. However, objects exist in the real world independently of whether they are observed or annotated by humans. Consequently, defining objects through the lens of human annotators introduces a layer of subjectivity and potential for error, as it depends on the annotator's interpretation of a static image to capture the essence of what objects are.

In contrast, humans do not require explicit guidance or direct supervision to learn what objects are, how they are bounded, and how to interact with them or understand their interaction with the world. This understanding develops naturally through experience, underscoring a fundamental difference between how humans and supervised computer vision systems approach object segmentation. Reflecting on our cognitive capabilities raises a question: What properties of an object enable humans – and potentially machines – to perceive objects?

1.2 Human Perception of Objects

Studies on object perception by Elisabeth Spelke [118, 119] suggest that infants can perceive and track objects *before* they learn to interact and manipulate them. As adults, when presented with a simple example of a scene with a block that serves an occluder, as follows:



we will likely conclude that there is a single object behind the occluder, following the cues of consistent texture, regular shape, or alignment of the visible parts.

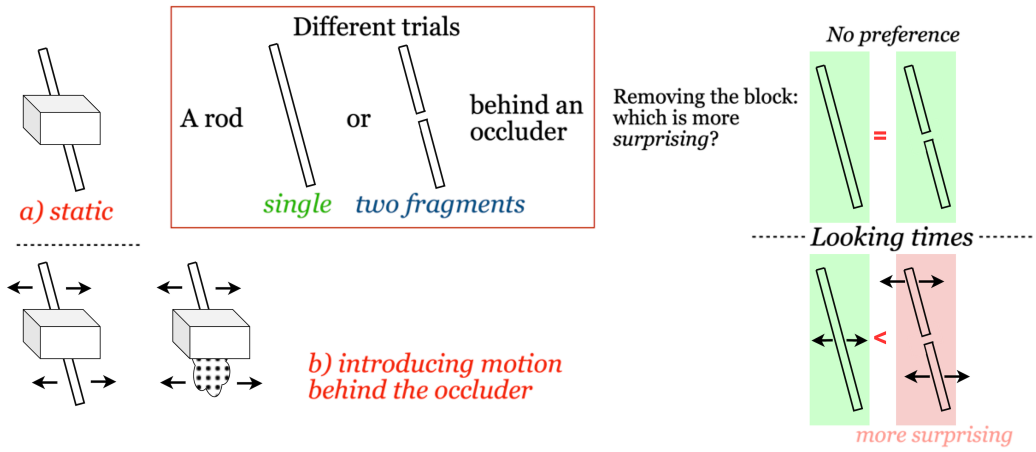


Figure 1.2: Illustration of the experiment studying object perception in infants [119]. A single or fragmented center-occluded object is presented to infants. Upon the removal of the occluder, the perception of an object is indeterminate between a connected object and two object parts (a). If motion is introduced to the object behind the occluder (b), infants perceive the moving parts as belonging to the same object, regardless of parts alignment, texture consistency or shape regularity.

Interestingly, research suggests that this tendency to group surfaces into the simplest possible forms through these static properties develops later in the life of an infant. Studies by Spelke analyzed how 4-month-old infants perceive objects under occlusion, before they can move or manipulate them. They were presented with different scenarios of the occluded scene from above, where behind the occluder block there was either a single rod or two separated rod pieces (see Figure 1.2 (a)). Upon removing the block, the level of surprise was measured by analyzing looking times at the object behind the occluder. It turns out that the looking times did not differ for both scenarios, indicating that infants did not have a preference for either the single or separated objects. Their perception appeared to be indeterminate between a connected object and two object fragments, regardless of static properties such as texture, shape, or alignment.

In another variant of the experiment, motion was introduced to the center-occluded object in both scenarios, before the occluder was removed (see Figure 1.2 (b)). In this setting, once the object was uncovered, infants that were shown the separated rod found the revelation much more surprising, indicating the preference for a single connected object. Infants perceived an occluded object as a connected unit when the parts of the object moved together behind the occluder. Any translation of the object in the three-dimensional space – lateral, vertical, and translation in depth – led the infants to perceive a continuous object. Moreover, this perception was not affected by

other object properties. Even when a center-occluded object was asymmetric and not uniform in texture and color, it was perceived as connected just as strongly as a simple shape with a uniform texture and color. In short, infants perceive parts of objects that move together as parts of the same object.

These experiments show how the cohesive motion of object parts provides a very strong cue to determine what an object is – stronger than texture, color, or shape regularity. More precisely, Spelke distinguished core principles that help explain how young children understand and perceive objects in their environment. *Cohesion* states that objects are perceived as single, bounded wholes, expected to move as a unit, maintaining its boundaries over time. *Continuity* suggests that objects move along smooth, uninterrupted paths, with infants anticipating predictable motion without spontaneous appearances or disappearances. *Contact* implies that objects do not exert influence unless they physically touch, leading infants to expect that movement or trajectory changes occur only upon collision. *Solidity* asserts that objects are solid and unyielding, incapable of moving through or sharing space with other solid objects. Finally, *Persistence* indicates that objects are perceived as continuously existing, even when out of sight, leading to an expectation of their constant presence in time and space.

Building on these foundational insights, there lies an intriguing possibility: leveraging these core principles of object perception for advancing unsupervised learning in object segmentation. By applying principles like cohesion, continuity, contact, solidity, and persistence, we could guide machine learning models to parse visual information in a more human-like manner, potentially reducing or even eliminating the need for precise segmentation annotation. This approach hints at a more natural, intuitive method of teaching AI to understand and interpret visual data, drawing inspiration from the earliest stages of human cognitive development.

1.3 Leveraging generative models

Generative models stand as powerful tools in the field of artificial intelligence and computer vision, capable of creating images that closely mimic the distribution of real image sets. Their utility extends beyond mere image generation; they can be efficiently used to guide image manipulation processes as well.

In the context of previously stated principles of object perception, it seems plausible to explore the use of generative models for simulating actions on objects within *static* images, where motion cannot be observed. This could be exemplified by manipulating an object's position within an image, similar to translating it in a three-dimensional space. Such manipulations not only test the model's understanding of object continuity and cohesion but also its ability to maintain realism in the context of physical laws.

Moreover, the ability of certain generative models to evaluate the realism of synthesized images could align with the cognitive processes humans experience in distinguishing between expected and unexpected changes in their visual environment. This aspect can be particularly useful in refining AI's ability to distinguish between realistic and unrealistic alterations in a visual scene, mirroring human perceptual development.

1.4 Thesis contributions

This thesis addresses the challenge of unsupervised object segmentation in images. We want to emphasize that despite talking about *moving* objects throughout the thesis, we develop methods that work on *static images*, not videos. Specifically, we focus on *salient objects*, typically the main focus or point of interest for the viewer, often associated with the image's *foreground*. We develop methods to segment such objects without relying on manual annotations – object labels, bounding boxes, landmarks, or pre-trained object detectors and classifiers – and train segmentation models using only collections of images. We present a method employing generative models to create a layered scene representation through the realism of manipulated generated images. This innovative approach enables the generation of separate background and foreground scene components. We demonstrate how this generative model can segment real images. Subsequently, we explore how these models can be simplified to output a segmentation map directly for an image. We illustrate the use of generated segments as pseudo-labels for object segmentation models and their effective translation to real-world images. Finally, we introduce a model that learns salient object segmentation maps directly from real images, utilizing methods to approximate manipulated image distributions from adversarial training. We build upon recent advanced self-supervised image models, showing their adaptability for powerful image representation and manipulation tasks, such as inpainting. Lastly, to advance research on the generative models used in this thesis, we propose a new training approach for Generative Adversarial Networks. This method incorporates a novel loss inspired by contrastive learning literature, taking the form of a statistical divergence between distributions. This loss offers improved gradients, enabling the active pursuit of missing modes in the generated data's distribution.

1.4.1 Chapter outline

Chapter 2: Background. We provide a general overview of object segmentation methods with a special focus on unsupervised object segmentation, which is the central theme of this thesis. This includes modern self-supervised models, which many segmentation methods are based on. Moreover, we introduce the Generative Adversarial Networks, used extensively in our methods.

Chapter 3: Emergence of Object Segmentation in Perturbed Generative Models. Building on the observation that the location of objects can be perturbed locally relative to a given background while maintaining the realism of a scene, we introduce a method to generate separated background and foreground object components of a scene. We train a generative model such that a composite image obtained by overlaying a shifted foreground on the generated background yields a realistic scene. Because the generator is unaware of the shifts in the image, it must produce layered representations that are realistic for any such random perturbation. Levering a layered *generative* model, allows to shift the foreground objects without the need for inpainting. Finally, we train an encoder to map real images into the generated layered representation with a corresponding mask for an object. This chapter corresponds to the NeurIPS publication [10].

Chapter 4: Unsupervised Learning of Object Segmentation From Perturbed Generative Models. We further explore the perturbed generative models. We show how they can be simplified and explore different ways of training a segmenter that produces segmentation maps for real images, introducing an end-to-end training procedure that can achieve that. We show that a segmenter trained on synthetic composite images and their corresponding generated masks works well on real data as well.

Chapter 5: MOVE: Unsupervised Movable Object Segmentation and Detection. We introduce a novel method to segment objects without any supervision. Contrary to previous approaches, we train a segmenter directly on real images utilizing recent state-of-the-art self-supervised features. While we still use adversarial training, we do not need to generate the images as we utilize a powerful inpainting network based on Masked Autoencoders. This chapter corresponds to the NeurIPS publication [11].

Chapter 6: Generative Adversarial Learning via Kernel Density Discrimination. Training Generative Adversarial Networks (GANs) involves maintaining a careful balance between the discriminator and generator networks. However, this process is susceptible to mode collapse, leading to less-than-optimal solutions. To overcome this, we propose a new GAN loss method called Kernel Density Discrimination (KDD GAN). This approach utilizes statistical divergence between kernel density estimates of real and generated data distributions in the feature space. This is effective even when the discriminator is not optimal. By doing so, we provide improved training gradients that encourage the generator to cover previously unrepresented modes in its distribution. The outcome is a notable enhancement in the quality of generative models. This chapter corresponds to the publication [69].

Chapter 2

Background

This chapter provides an overview of the fundamental concepts and techniques in object segmentation and unsupervised learning, with an emphasis on methods that are relevant to this thesis. It is noteworthy that supervised image segmentation methods have been thoroughly explored for their straightforward approach in associating input images with pixel-precise labels. However, advancements in this field also bear significance for the unsupervised segmentation domain, which is the central interest of this thesis. Specifically, innovations in neural network architectures, designed to produce image masks, are potentially beneficial in an unsupervised context, despite the lack of a direct supervisory signal.

In recent years, unsupervised learning has seen substantial advancements, driven by developments in generative models and self-supervised learning strategies. These approaches have begun to close the gap with their supervised counterparts, offering new ways to leverage the abundance of unlabeled visual data. The focus here is on how generative models can be used to separate real or generated images into distinct background and foreground components, enabling effective object segmentation without explicit labels. Furthermore, self-supervised learning, by utilizing inherent data characteristics as proxy labels, has emerged as a powerful tool for learning useful representations from unlabeled data. This chapter provides an overview of these approaches, focusing on their application for image segmentation.

2.1 Image Segmentation Formulation

The task of image segmentation involves partitioning an image into multiple segments that can represent semantic regions, objects, parts, or boundaries within the image. More formally, the image segmentation task can be formulated as follows. Given an input image I of size $H \times W$ (where H and W represent the image height and width, respectively), the goal is to produce a segmentation map S of the same size, where

each pixel in S corresponds to a semantic label or indicates the presence of an object. Therefore, the aim is to learn a mapping

$$F : I \rightarrow S, \tag{2.1}$$

based on a training dataset. This task is particularly challenging due to variations in object appearances, scale, and occlusions, as well as the need for precise boundary delineation.

Depending on the specific image segmentation task, pixels in S can take binary values (*e.g.*, for salient, or foreground object segmentation; see Figure 1.1) or a form of one-hot encoded vector, representing belonging to one of many classes (*e.g.*, for semantic, instance or panoptic segmentation), among others. In this thesis, we focus on unsupervised salient/foreground object segmentation.

In the case of supervised image segmentation, a labeled dataset is available, where each training example consists of an input image and the corresponding output segmentation map. Therefore, the mapping function $F : I \rightarrow S$ is learned based on a dataset of image-segmentation pairs $\{(I_i, S_i)\}$, where i denotes the i-th sample in the dataset, and is often cast as a pixel classification problem. In contrast, in the case of unsupervised segmentation, which is the topic of this thesis, the only data available is the collection of images $\{I_i\}$.

2.2 Neural Networks for Image Segmentation

Neural networks have revolutionized the field of image segmentation by providing powerful tools for learning complex patterns and representations from visual data. This section outlines the evolution of neural network architectures in the domain of image segmentation.

2.2.1 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) [67] are the backbone of many image analysis tasks. In the context of image segmentation, CNNs excel at extracting hierarchical features from images, which are crucial for distinguishing between different semantic regions. Early architectures like FCN (Fully Convolutional Network) [81] laid the groundwork by demonstrating that a CNN, traditionally used for image classification, could be adopted for pixel-wise predictions necessary for segmentation tasks.

2.2.2 Encoder-Decoder Architectures

The encoder-decoder structure is a significant advancement in segmentation networks, designed to efficiently capture context and spatial information. The encoder progressively reduces the spatial dimensions of the input image, abstracting high-level

Figure 2.1: U-Net architecture illustration (with 32×32 pixels at the lowest resolution). Each blue rectangle represents a multi-channel feature map, with the number of channels indicated above it. The dimensions in the x-y plane are noted in the bottom left corner of the rectangle. White rectangles symbolize duplicated feature maps, and the arrows illustrate the various operations involved. *Link to the figure:* [107], Fig. 1.

semantic information. In contrast, the decoder part reconstructs the segmentation map from the encoded features, progressively increasing the resolution to achieve pixel-level precision [94]. U-Net [107], a notable example, introduced a symmetric architecture with skip connections that allow the flow of information between encoder and decoder layers (see Figure 2.1), significantly improving the accuracy of segmentation.

2.2.3 Instance Segmentation with Mask R-CNN

Mask R-CNN [42] extends the Faster R-CNN [106] object detection framework to address instance segmentation. It combines CNNs with a region proposal method for instance segmentation, effectively identifying and segmenting individual objects in images. A mask prediction branch is added to the existing structure for object detection, enabling simultaneous detection and segmentation of objects at the instance level. This approach allows Mask R-CNN to output a binary mask for each instance in addition to the class and bounding box. Mask R-CNN demonstrates high efficiency and accuracy in instance segmentation tasks, making it a significant model in the evolution of neural networks for image segmentation.

2.2.4 Attention Mechanisms and Transformer Models

Attention mechanism, and more recently, Transformer models, have introduced a paradigm shift across various domains of machine learning. Unlike traditional CNNs, which process image regions with a fixed receptive field, attention-based models such as the Vision Transformer (ViT) [28] use attention mechanism [125] to selectively focus on different parts of the image. This approach allows the model to weigh the importance of each part of the image according to the task at hand, enabling the modeling of long-range dependencies and interactions between image patches. In the context of image analysis, this capability is particularly beneficial for understanding complex scenes and defining object boundaries more accurately. An example of leveraging Transformer architecture for image segmentation is Maskformer [21], which combines Transformers with a per-pixel mask loss and per-mask classification loss to efficiently segment objects and scenes.

2.3 Unsupervised Object Segmentation Methods

Unsupervised object segmentation methods play a crucial role when labeled data is unavailable. These techniques focus on identifying and segmenting objects within images by relying on the inherent structure and distribution of the data itself. This section introduces various groups of unsupervised segmentation methods, each employing different strategies to achieve segmentation without the need for manually annotated data. We explore methods that utilize clustering algorithms, generative models, and self-supervised learning techniques.

2.3.1 Clustering and Hand-crafted Methods

To avoid manual annotation, the task of interest can be cast in the unsupervised learning framework (see [6, 33]). Early fully unsupervised methods for segmentation relied on a form of clustering of color, brightness, local texture, or some feature encoding. For instance, the superpixel clustering method of [1] and the mean-shift method of [23] are some of the first segmentation approaches based on prescribed low-level statistics.

Several hand-crafted methods have been proposed to address the task of unsupervised segmentation, particularly focusing on saliency detection, by leveraging human-designed image features. To name a few, the discriminative regional feature integration approach of [52] utilizes color contrast, texture, and boundary-based features to distinguish salient objects. In a similar vein, the optimization technique for saliency detection by [156] leverages spatial coherence and edge density as robust background indicators. The method proposed by [74] for saliency detection relies on dense and sparse reconstruction, using features like color uniqueness and spatial distribution to highlight salient regions. Another approach by [51] employs Markov chains, constructing a graph with edges defined by features such as color similarity and spatial proximity to estimate saliency. The weighted sparse coding framework for saliency detection [73] utilizes sparsity in feature representation, emphasizing contrast and uniqueness. Hierarchical saliency detection by [138] incorporates multi-scale image features, including intensity, color, and orientation, to progressively refine saliency maps. The work of [158] introduces hierarchy-associated rich features, combining color, texture, and location information to effectively detect salient objects. Lastly, [92] combines different hand-crafted methods into a deep learning framework.

2.3.2 Mutual Information and Scene Decomposition

Unsupervised segmentation can also be formulated as a pixel-wise image partitioning task. IIC [50] defines the task as a classification problem with a known number of segment types, such that the mutual information between the predicted partitions of transformed versions of the same image is maximized. In [98] they learn image

clustering by maximizing mutual information between different orderings of pixels in autoregressive models. CIS [141] also utilizes mutual information between segments, but due to the use of optical flow is only applicable to videos. The work of [110] points out that the correct mask maximizes the inpainting error for both the background and the foreground; however, for complex real-world datasets, there is too much ambiguity in selecting such regions. Several methods for unsupervised scene decomposition were proposed, including [15, 30] and [80] that use spatial or slot attention and [36–38] that model images as a spatial mixture model to perform unsupervised segmentation. However, these approaches have only been shown to work on simpler synthetic or controlled datasets.

2.3.3 Generative Methods

A wide range of methods exploits generative models trained without supervision to find a segmentation mask or to generate synthetic data to train a segmenter. In one line of work, it is observed that a pre-trained generator of large-scale Generative Adversarial Networks (GANs; see section 2.4) [35] contains directions in the latent space that modify the foreground and the background independently, while keeping the content of the image. The work of [129] finds the latent shift that acts on each pixel as one of two affine operators, while [85] finds the directions that preserve the edges while modifying the brightness.

Other methods exploit scene compositionality to generate or decompose the scene into a layered representation. The work of [124] proposes a GAN that generates a background and individual objects by modeling their relational structure with attention, however, the method is shown to work only on relatively simple datasets. In [65] they generate parts of an image with a GAN and RNN and apply restrictions on alpha channels to avoid degenerate solutions, but their blending procedure does not encourage resulting composite images to necessarily contain the exact generated parts. The method of [140] uses GAN and LSTM to generate a background, a foreground with a mask, and a transformation matrix to learn where to place the objects with a spatial transformer. More in general, generative methods make different assumptions on the information in the object segment. For example, [18] assumes that only when the segmentation is correct, a generative model can replace the original object with another in a realistic manner. In [8, 97] they use generative models to solve several tasks simultaneously, including foreground segmentation, but they rely on the assumption that a dataset of only backgrounds is available. The works of [3, 105] build on the idea that realistic segmentation masks would allow the copy-pasting of a segment from one region of an image to another. They do, however, either use pre-trained object detectors as weak supervision [105] or do not guarantee that the copied object retains a realistic context.

Figure 2.2: The Masked Autoencoder (MAE) method uses a pre-training approach where around 75% of image patches are randomly masked. An encoder processes the visible patches, and after encoding, mask tokens are added. A small decoder then reconstructs the original image. After pre-training, the decoder is removed, and the encoder is used on full images for recognition tasks. *Link to the figure:* [44], Fig. 1.

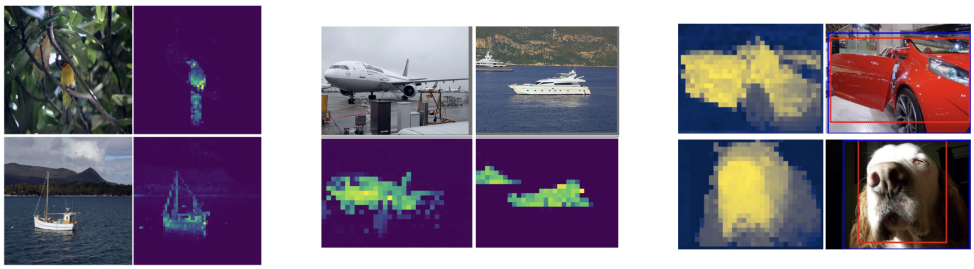
All these methods require either additional assumptions, were shown to work only on simple or synthetic datasets, or require some sort of additional supervision.

2.3.4 Self-Supervised Pre-trained Models

Self-supervised learning (SSL) has gained increased attention in the machine learning community, particularly within the domain of computer vision. It enables models to learn rich representation from unlabeled data by predicting part of the input from other parts or by solving auxiliary tasks. For instance, in the context of images, earlier deep learning SSL models learn a feature representation through autoencoding [46], predicting the color of a grayscale image [151], solving a Jigsaw puzzle [95] or predicting the rotation of an image [34]. Such representations, learned in an unsupervised way, may be used to discover semantic data structure through clustering or finetuned for downstream tasks, like image classification, object detection or segmentation.

In recent years, SSL methods have gained a significant improvement, mainly driven by the success of contrastive learning methods [16, 19, 20, 43] and scalability of Vision Transformers [28]. Contrastive learning aims to learn representations by maximizing the agreement between the representation of differently augmented views of the same datapoints while minimizing the agreement between views of different data points. Another approach, Masked Autoencoders (MAE) [44], applies masked modeling in vision domain, inspired by the success of masked language models in natural language processing. During the training a significant portion of the input image is randomly masked and the objective is to reconstruct the masked parts (see Figure 2.2).

In the context of unsupervised image segmentation, self-supervised methods have attracted increasing attention due to emerging properties when such models are trained at scale. Most prominently, DINO [17] is an SSL model that utilizes knowledge distillation between a student and a teacher network, both processing different augmentations of the same image, to learn rich visual representations without labeled data. It has been shown that the attention maps of the pre-trained DINO Vision Transformer can group semantically meaningful regions within images, despite not using any labels or having any direct incentive to do so. Multiple works followed that direction, finding that DINO feature representation provides a good baseline to obtain unsupervised image segmentation. The work of [2] explores the applicability of such features for



DINO [17] LOST [115] TokenCut [136]

Figure 2.3: Examples of different applications of DINO features and attention maps to segment objects.

co-segmentation and finding correspondences through feature clustering. LOST [115] designs a seed expansion strategy to obtain an object segment. DeepSpectral [84] and TokenCut [136] propose a segmentation method based on solving the graph-cut problem with spectral clustering (see Figure 2.3 for examples). *Semantic* segmentation can also be achieved by clustering features across the dataset, as shown in DeepSpectral [84] and STEGO [40], although careful dataset-dependent tuning might be needed for these methods. Beyond DINO, FreeSOLO [135] uses DenseCL features [134] to obtain coarse object masks. SelfMask [114] proposes a clustering approach that can use multiple SSL features and evaluates all possible combinations of DINO [17], SwAV [16] and MOCOV2 [43].

2.4 Generative Adversarial Networks

Generative Adversarial Networks (GANs) [35] were a breakthrough in generative modeling, where the goal is to model the data distribution. GANs aim to solve the problem of generating new data points that are indistinguishable from real data. They achieve this through an adversarial framework involving two competing networks: the generator, which creates data resembling the real distribution, and the discriminator, which evaluates how well the generated data matches the statistical properties of the real data distribution. Generative learning finds applications in many computer vision tasks, such as image translation [29, 48, 100, 155], image processing [64, 68], image restoration [99, 123, 151], text to image mapping [70, 103, 104, 147] and, more in general, to define image priors in image-based optimization problems [86, 123].

2.4.1 Problem Formulation

The goal of Generative Adversarial Networks is to find a mapping from a known distribution, typically a multi-variate Gaussian $p_z(z)$, to the data space $p_{data}(x)$. Their

training relies on an adversarial min-max game, in which two neural networks, a generator (G) and a discriminator (D), are trained against each other in a zero-sum game. The discriminator is trained to distinguish real samples x from fake ones $G(z)$ synthesized by the generator, while the generator is trained to fool the discriminator. More formally, this adversarial process can be described by the following value function $V(G, D)$:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.2}$$

Here, x represents real data drawn from the data distribution $p_{data}(x)$, and z represents input noise variables drawn from a distribution $p_z(z)$ used by the generator to produce data. The discriminator outputs a scalar $D(x)$ linked to the probability that x came from the real data rather than the generator. The generator aims to produce data $G(z)$ as realistic as possible to fool the discriminator.

The goal of the discriminator is to maximize $V(D, G)$ for a given G, meaning it tries to assign the correct labels to both real and generated data. Conversely, the generator aims to minimize $V(D, G)$ given the discriminator D, meaning it tries to produce data that the current discriminator will mistakenly classify as real. This formulation involves the adversarial training process, where both G and D improve through iterations. The generator learns to produce more accurate representations of the data distribution, while the discriminator becomes better at identifying subtle differences between real and generated data. The equilibrium of this game is reached when G generates data indistinguishable from real data, which corresponds to a known statistical divergence, *e.g.*, the Jensen-Shannon Divergence (JSD) [35].

2.4.2 Limitations and Advancements

Although GANs offer numerous benefits, training them poses a lot of challenges in practice. The stability of the training process relies on finding a balance between the two networks, ensuring that neither becomes too powerful too quickly, to maintain the adversarial dynamic. GANs are also prone to mode collapse, where the model only captures a subset of the data distribution and fails to represent its full diversity.

GANs have continually evolved and improved through various iterations since they were first introduced. Works such as [4] and [96] focus on training the generator to minimize other statistical divergences that exhibit better properties compared to the JSD in the original work. A variety of other loss functions and GAN frameworks have been introduced, claiming better stability and quality of generated samples, *e.g.*, [9, 39, 82, 148] One complementary line of research explores additional regularization terms such as using a gradient penalty for the discriminator [87], consistency regularization [149] or differentiable augmentations [153] to various degrees of success. Many works

aim to address the mode collapse problem, including mini-batch discrimination and label smoothing [108], UnrolledGAN [88] and PacGAN [78]. Another line of work introduces image GAN models that process progressively larger images and grow in size. Progressive GAN [54] and StyleGAN [55, 57] proved to work particularly well to improve the stability and quality of generated images.

Training GANs on large-scale datasets is a challenging task. State of the art models such as BigGAN [14] require a substantial amount of compute resources. Moreover, many of them require a post-hoc processing to reduce spurious samples. [27] proposes to tackle both issues by filtering the dataset using instance selection. They argue that the model's capacity is wasted on low density regions of the empirical distributions of the data. Their results show that instance selection enables the training of better GAN models with substantially fewer parameters and less training time.

A recent addition to this list are methods that capture more structure into the latent representation of the discriminator through the use of Contrastive Learning [49, 53, 145]. One such example is ContraGAN [53], where the authors introduce a new regularization term, called 2C loss, based on the NT-Xent loss [19] used commonly in Contrastive Learning. The introduced loss term aims at capturing the data-to-data and data-to-class relations in the dataset.

Chapter 3

Emergence of Object Segmentation in Perturbed Generative Models

In this chapter, we introduce a framework to learn object segmentation from a collection of images without any manual annotation. Since we cannot rely on the provided labels, we employ object properties that can help to define the object's location. Namely, we build on the observation that the location of object segments can be perturbed locally relative to a given background without affecting the realism of a scene. In fact, if object segments include some of the background, a small shift would reveal an unnatural edge with the background. Similarly, if the object segments do not include the entirety of the object, a small shift would reveal an unnatural-looking object occlusion or discontinuation. This is related to the cohesive motion principle of object perception described in Chapter 1.

Our approach is to first identify a powerful and general principle to define what an object segment is and then to devise a model and training scheme to learn through that principle. With reference to Figure 3.1, we propose to build a generative model that outputs a background image, a foreground object, and a foreground mask. This model is trained in an adversarial manner against a discriminator. The discriminator aims to distinguish the composite image, obtained by overlaying the output triplet of the generator, from real images. This training alone provides no incentive for the generator to produce triplets with correct object segmentations. In fact, a trivial solution is to have the same realistic image for the foreground and background and a random mask (see Figure 3.2, first row). To address this failure, our framework introduces the concept of learning through the perturbation of the model output. According to our object segment definition, we could introduce a small random shift between the

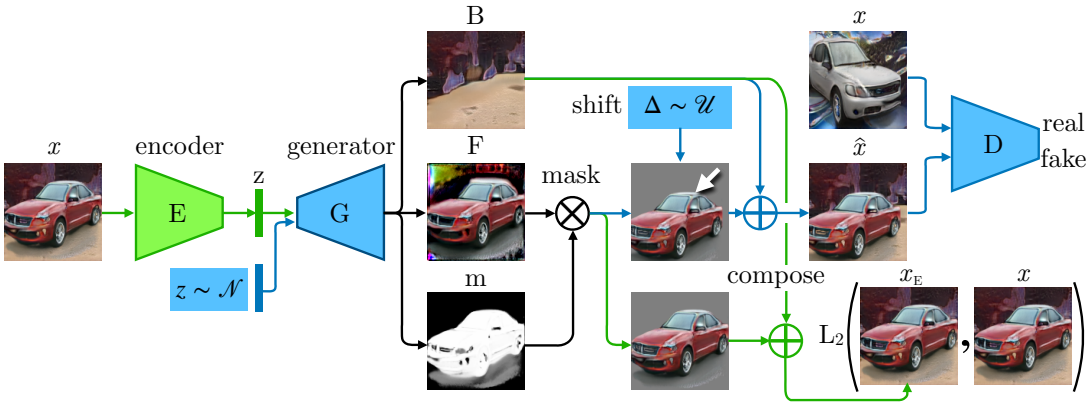


Figure 3.1: Illustration of the proposed architecture to learn to generate realistic layered scene representations through G (blue path) and to learn to map images to a layered representation through E (and G), *i.e.*, to segment objects (green path). The layered representation consists of three components: 1) a background image B, 2) a foreground image F, and 3) a(n alpha matte) mask image m. A crucial component of our model is the generation of random shifts p of the foreground object (in particular, such that they are independent of the input vector z to G) during the training of the generator. The generator is trained adversarially against a discriminator D. Once the generator G is trained, the encoder E can be trained to extract z, which encodes the layered representation.

foreground and background outputs and still obtain a realistic composite image. If this perturbation is unknown to the generator before producing its triplet, then it is forced to output realistic object segments (see Figure 3.2, last row). As a separate step to retrieve the segmentation of an image, we propose to train an encoder network. The encoder is paired with the generator so as to form an autoencoder. The encoder maps an image to a feature vector, which is fed as input to the generator so that it outputs a triplet that can be used to rebuild the input image. The combination of both of these steps allows us to train the encoder to detect and segment objects in images without any manual annotation (see the green path in Figure 3.1).

Contributions. We introduce a fully unsupervised learning approach to segment objects. Unlike in prior work, we do not make use of object detectors, classifiers, bounding boxes, landmarks, or pre-trained networks. To our knowledge, this is the first such solution working on complex real images. Moreover, the proposed method is quite general as we demonstrate it on several object categories qualitatively, and quantitatively on the LSUN Car dataset [143] with Mask R-CNN [42] used as ground truth as well as CUB-200 dataset with provided annotations. Although we evaluate our approach on a dataset with a single object category at a time, our framework can

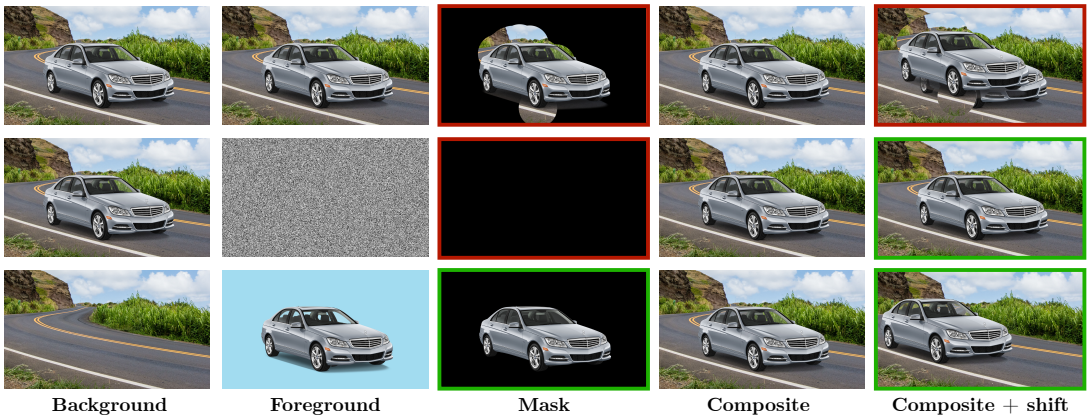


Figure 3.2: **First row**: Trivial solution, where the background and the foreground are identical and any mask produces a valid composite scene. However, a random foreground shift reveals the invalid segmentation. **Second row**: Trivial solution, where the whole scene is generated in the background and the mask is always empty. **Last row**: The scene after a random shift is valid only when the background generation and the object segmentation are valid and the mask is not empty.

potentially work on mixed object collections (see Figure 3.5). We use single object category datasets because of current GAN limitations.

3.1 Related Work

In Chapter 2 we introduce different unsupervised methods for object segmentation. Most of the methods prior to the approach described in this chapter work either on simple datasets or utilize some sort of weak supervision, for example, object bounding boxes or pre-trained supervised models. In contrast, this method works on real-world images and does not make use of any form of supervision or pre-trained models.

The work that most closely relates to ours is by [105]. In this paper, the authors build on the idea that realistic segmentation masks would allow the copy-pasting of a segment from one region of an image to another. This remarkable principle can be used to define what an object is. More generally, one could say that pixels belonging to the same object should be more correlated than pixels across objects (including the background as an object). The weak correlation between the object and background is what allows introducing a shift without compromising the realism of the scene. However, the weak object-background correlation means also that not all shifts yield plausible scenes. This is why [105] study the object placement and introduce some randomized heuristics as approximate solutions. In contrast, in our work, we avoid heuristics by noticing that small shifts are almost always valid. The price to pay is that

background inpainting is required. That is why we introduce a generative model that learns to output a background and a foreground image in addition to the segmentation mask. One important aspect of the design of unsupervised learning methods is to avoid degenerate solutions. [105] build a compositional image generator as done by [97] and then train a segmenter adversarially to a discriminator that classifies images as realistic or not. A degenerate solution for the segmenter is to avoid any segmentation, as the background looks already realistic. The authors describe two ways to avoid this scenario: One is that the dataset of real images contains objects of interest and therefore an empty background would be easily detected by the discriminator. The second is that a classification loss (pre-trained on object identities) would ensure that an object is present in the composite scene. This approach assumes some knowledge about objects (*e.g.*, where they are) and works well on relatively small images (28×28 pixels). In contrast, our approach does not require such assumptions and we show its performance on (relatively) high resolution images. In our approach we require that the mask has a minimum number of non-zero pixels, *i.e.*, we learn to generate segments with a minimum size (this avoids the degeneracy illustrated in the second row of Figure 3.2). This is not a restriction, because we are not making assumptions on single images, but, rather, on the distribution of the image dataset. Then, we establish the correspondence between images and segments in a second step where we train an encoder network. The encoder learns to map images to a suitable noise vector for the pre-trained generator, such that it outputs background, foreground, and mask that autoencode the input image after composition (see Figure 3.1). The design of such generative models is only possible today thanks to the progress driven by the latest Generative Adversarial Networks of [55], mentioned in Section 2.4, page 29, which we exploit in this work.

3.2 Learning to Segment without Supervision

Our approach is based on two main building blocks: A generator G and an encoder E (see Figure 3.1 for an illustration of the proposed method). The generator is trained against a discriminator in an adversarial manner with the latest high-quality StyleGAN (Generative Adversarial Network) by [54, 55]. G learns to generate composite scene samples to the extent that the discriminator cannot distinguish them from real images. There are several important aspects that we would like to highlight. Firstly, the training requires no correspondence between the real images and the generated scenes. It allows us to impose constraints on the average type of generated scenes we are interested in, rather than a per-sample constraint. For example, we expect the average scene to have an object with a support of at least $15\% - 25\%$ of the image domain, a condition that may not hold in each sample. Secondly, during training, we introduce a random

shift unknown to the generator. Thus, the generator must output a background and a foreground that can be combined with arbitrary small relative shifts and still fool the discriminator into believing that the composite image is realistic. This is an implicit way to define what an object is, that avoids manual labeling altogether. The second building block in our approach is an encoder E that learns to segment images. The encoder followed by the generator and the image composition form an autoencoder. The encoder E maps a single image x to a feature vector z that, once fed through the generator, yields its background B (with inpainting), its foreground object F, and its foreground object mask m. The correspondence between images and their object segmentation is thus obtained through the training of the encoder. In the following sections, we explain our approach more in detail.

3.2.1 A Generative Model of Layered Scenes

Consider an $M \times N$ discrete image domain $\Omega \subset \mathbb{Z}^2$. In our notation, we consider only grayscale images for simplicity, but in the implementation we work with color images. We define the representation of a scene as a layered composition of 2 elements: a background image $\mathrm{B} : \Omega \mapsto \mathbb{R}$ and a foreground image $\mathrm{F} : \Omega \mapsto \mathbb{R}$. Although the foreground is defined everywhere, it is masked with an alpha matte $\mathrm{m} : \Omega \mapsto [0, 1]$ in the image composition. The composite image $\bar{x} : \Omega \mapsto \mathbb{R}$ is then defined at each pixel $\mathbf{p} \in \Omega$ as

$$\bar{x}[\mathbf{p}] = (1 - \mathrm{m}[\mathbf{p}])\mathrm{B}[\mathbf{p}] + \mathrm{m}[\mathbf{p}]\mathrm{F}[\mathbf{p}]. \tag{3.1}$$

We define a generator $\mathrm{G} : \mathbb{R}^k \mapsto \mathbb{R}^{M \times N}$ through a convolutional neural network (as described in [55]) such that, given a k-dimensional input vector $\mathrm{z} \sim \mathcal{N}(0, I_d)$, it outputs three components $\mathrm{G}(\mathrm{z}) = [\mathrm{G_B}(\mathrm{z}), \mathrm{G_F}(\mathrm{z}), \mathrm{G_m}(\mathrm{z})]$, where $\mathrm{G_B}(\mathrm{z}) = \mathrm{B}$, $\mathrm{G_F}(\mathrm{z}) = \mathrm{F}$ and $\mathrm{G_m}(\mathrm{z}) = \mathrm{m}$. The generator is then trained in an adversarial manner against a discriminator neural network $\mathrm{D} : \mathbb{R}^{M \times N} \mapsto \mathbb{R}$. Our implementation is based on StyleGAN, which, in turn, is based on the Progressive GAN of [54], a formulation using the Sliced Wasserstein Distance.

3.2.2 Learning through Model Perturbation

If we trained the generator with fake images according to eq. (3.1) and we assumed a perfect training, the learned model could be a trivial solution, where the background $\mathrm{G_B}(\mathrm{z})$ and the foreground $\mathrm{G_F}(\mathrm{z})$ are identical and realistic images, and the mask $\mathrm{G_m}(\mathrm{z})$ is arbitrary (see Figure 3.2, first row). In fact, there is no incentive for the generator to learn anything more complex than that, and, in particular, to associate foreground objects to the foreground mapping $\mathrm{G_F}(\mathrm{z})$. Even the constraint that the average value of the segments $\mathrm{G_m}(\mathrm{z})$ should be at least $15\% - 25\%$ is not sufficient to make the generator mapping more meaningful. We use the constraint that foreground objects

can be translated by an arbitrary small shift $\Delta \sim \mathcal{U}([-\delta, \delta] \times [-\delta, \delta])$, with δ a given range of local shifts, and would yield still a realistic composite image. Formally, this can be written by updating eq. (3.1) as

$$\hat{x}[\mathbf{p}] = (1 - \mathrm{m}[\mathbf{p} + \Delta])\mathrm{B}[\mathbf{p}] + \mathrm{m}[\mathbf{p} + \Delta]\mathrm{F}[\mathbf{p} + \Delta]. \tag{3.2}$$

Now, the generator has no incentive to generate identical foreground and background images, as a random shift would be immediately detected by the discriminator as unrealistic. Vice versa, it has an incentive to output foreground images and masks that include full objects. If the segments included some background or missed part of the foreground, a small random shift Δ would also yield an unnatural-looking image that the discriminator can easily detect (in particular, at the segment boundary; see Figure 3.2, last column). Therefore, now the generator has an incentive to output meaningful object segments. To make sure that the mask is non-empty, we impose a hinge loss on the average mask value

$$\mathcal{L}_{\text{size}} = \mathbb{E}_{\mathrm{z} \sim \mathcal{N}(0, I_d)} \left[\max \left\{ 0, \eta - {}^1\!/_{MN} |\mathrm{G_m}(z)|_1 \right\} \right] \tag{3.3}$$

with a mask size parameter $\eta > 0$ and also use a loss that encourages the binarization of the mask

$$\mathcal{L}_{\text{binary}} = \mathbb{E}_{\mathrm{z} \sim \mathcal{N}(0, I_d)} \left[\min \left\{ \mathrm{G_m}(z), 1 - \mathrm{G_m}(z) \right\} \right] \tag{3.4}$$

Finally, to train the generator we minimize the following loss with respect to $\mathrm{G_B}$, $\mathrm{G_F}$ and $\mathrm{G_m}$

$$\mathcal{L}_{\text{gen}} = -\mathbb{E}_{\hat{x} \sim p_{\tilde{x}}}[\mathrm{D}(\hat{x})] + \gamma_1 \mathcal{L}_{\text{size}} + \gamma_2 \mathcal{L}_{\text{binary}} \tag{3.5}$$

with $\gamma_1, \gamma_2 > 0$. To train the discriminator we minimize the following loss with respect to D

$$\mathcal{L}_{\text{disc}} = \mathbb{E}_{\hat{x} \sim p_{\tilde{x}}}[\mathrm{D}(\hat{x})] - \mathbb{E}_{x \sim p_x}[\mathrm{D}(x)] + \lambda \mathbb{E}_{\tilde{x} \sim p_{\tilde{x}}}[(|\nabla_{\tilde{x}} \mathrm{D}(\tilde{x})|_2 - 1)^2] + \epsilon \mathbb{E}_{x \sim p_x}[\mathrm{D}(x)^2] \tag{3.6}$$

where p_x is the probability density function of real images x, we define $\tilde{x} = \zeta x + (1 - \zeta)\hat{x}$ with random $\zeta \in [0, 1]$, $\lambda > 0$ is the gradient penalty strength and $\epsilon > 0$ prevents the discriminator output from drifting to large values, following [39] and [54].

3.2.3 Object Segmentation via Autoencoding Constraints

Once the generator has been trained, we can learn to associate background, foreground, and segments to each image. To do that, we can train an encoder E such that it retrieves, through the generator G, a composite image that matches the original input (see Figure 3.1, green path). The encoder $\mathrm{E} : \mathbb{R}^{M \times N} \mapsto \mathbb{R}^k$ maps x to $\mathrm{E}(x) \in \mathbb{R}^k$. Let us define $x_{\mathrm{E}} \doteq \left(1 - \mathrm{G_m}\left(\mathrm{E}(x)\right)\right) \odot \mathrm{G_B}\left(\mathrm{E}(x)\right) + \mathrm{G_m}\left(\mathrm{E}(x)\right) \odot \mathrm{G_F}\left(\mathrm{E}(x)\right)$. The loss used to train the encoder E can be written as

$$\mathcal{L}_{\text{auto}} = \mathbb{E}_{x \sim p_x} \left| x_{\mathrm{E}} - x \right|_1 + \mathbb{E}_{x \sim p_x} \left| \mathrm{D_{feat}}(x_{\mathrm{E}}) - \mathrm{D_{feat}}(x) \right|_2^2, \tag{3.7}$$

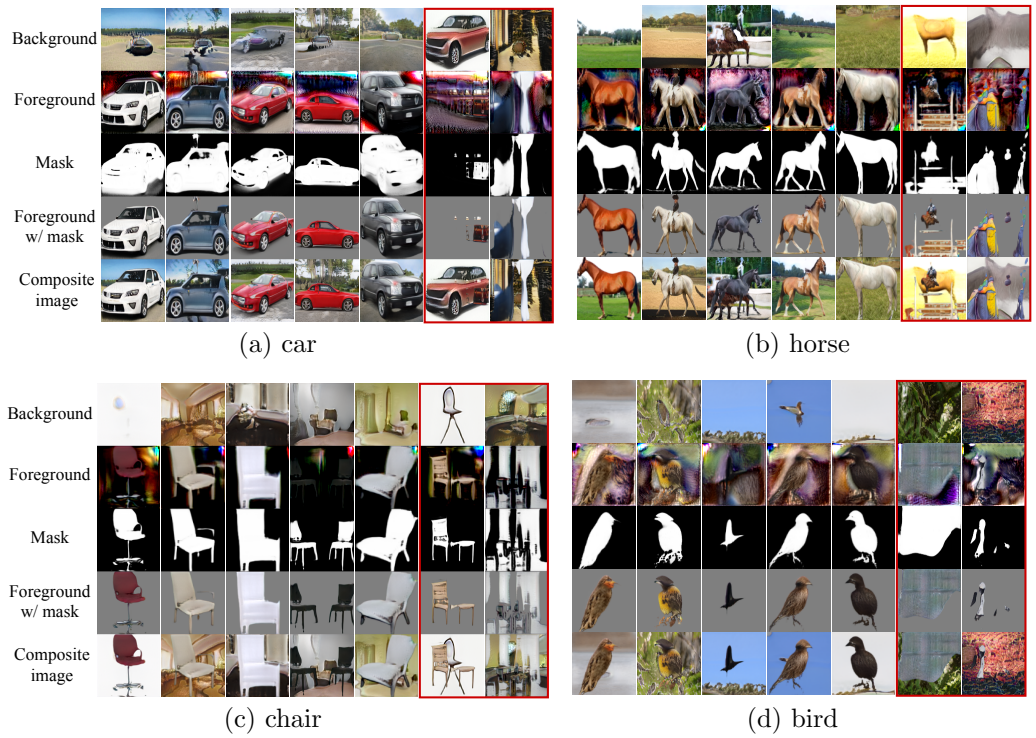


Figure 3.3: Generated 128 × 128 pixels backgrounds, foregrounds, masks, foregrounds with mask applied and composite images for 4 different image categories. Last two columns in each category show generator failures, *e.g.*, an object in the background or an unrealistic foreground.

where the second term is a perceptual loss that uses features from the trained StyleGAN discriminator.

3.3 Implementation

Experimentally, we find that current GAN methods are not yet capable of generating high-quality images from datasets of multiple categories (in a fully unsupervised manner). Thus, we mainly demonstrate our method on datasets with single categories. For all experiments, all the network architectures and details follow the StyleGAN [55] if not specified in this section. We use 2 separate generators, one outputs a 3 color channel background, while the other one has two outputs: a 3 color channel foreground and a 1 channel mask followed by a sigmoid activation function. Both generators take the same 512 dimensional Gaussian latent codes as input. We use mixing with a probability of 0.9 and feed two latent codes to two parts of the generator, split by a

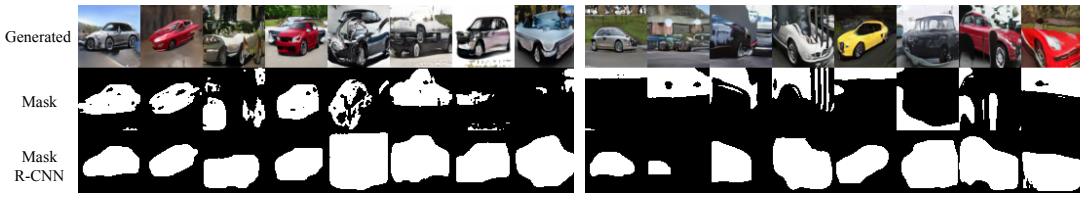
Table 3.1: Ablation study for the LSUN Car dataset. The IoU is computed using Mask R-CNN generated segmentations as ground truth. The reference IoU is computed using masks covering the entire image as segmentation.

Setting	**64 x 64 pixels**			**128 x 128 pixels**		
	IoU	reference IoU	detected cars	IoU	reference IoU	detected cars
(a) Default parameters	0.685	0.440	6293	0.533	0.432	7090
(b) No shift ($\delta = 0$)	0.039	0.428	6738	0.025	0.419	7578
(c) 25% shift ($\delta = 0.25 \cdot \text{size}$)	0.144	0.434	6493	0.094	0.426	7259
(d) Bg contrast jitter	0.765	0.454	6089	0.673	0.436	7046
(e) No random crops	0.264	0.374	6339	0.136	0.365	7520
(f) Mask size $\gamma_1 = 10.0$	0.733	0.443	6245	0.643	0.430	7241
(g) Min. mask size $\eta = 5\%$	0.693	0.458	6202	0.552	0.430	7256
(h) Single generator	0.550	0.446	6903	0.484	0.435	7544

randomly selected crossover point. We start the training with an initial resolution of 8×8 pixels and use progressive training to up to 128×128 pixels. We train with batch sizes 256, 128, 64, 32 and 32 for resolutions 8×8, 16×16, 32×32, 64×64 and 128×128 respectively. For each scale the number of iterations is set to process 1,200,000 real images. The local shift range δ described in Section 3.2.2 is resolution-dependent and set to $\delta = 0.125 \times$ `resolution`. For each resolution of training StyleGAN we first resize the real image to a square image of size $1.125 \times$ `resolution` and then take a random crop of size `resolution` to match the shifts in the generated data. We train the StyleGAN network on real images x and composite generated images $\hat{x}$ (eq. (3.2)) by alternatively minimizing the discriminator loss (eq. (3.6)) and the generator loss (eq. (3.5)). We set the discriminator loss parameters to $\lambda = 10$ and $\epsilon = 0.001$. In the generator loss we set $\gamma_1 = 2$ for the minimum mask size term and $\gamma_2 = 2$ for the binarization term. We optimize our GAN with the Adam optimizer ([61]) and parameters $\beta_1 = 0$, $\beta_2 = 0.99$. We use a fixed learning rate of 0.001 for all scales except for 128×128 pixels, where we use 0.0015.

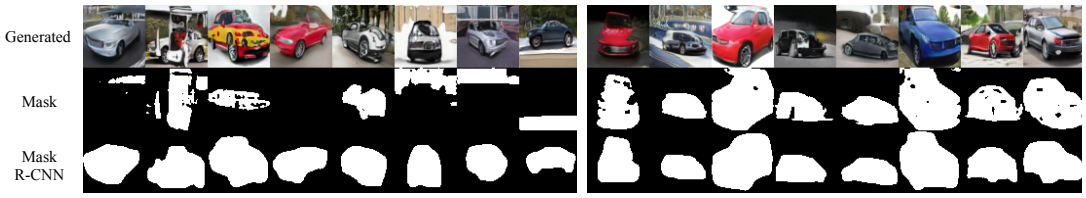
3.4 Experiments

We train our generative model on 4 LSUN object categories ([143]): `car`, `horse`, `chair`, `bird`. For each dataset we use the first 100,000 images. Objects in the datasets show large variability in position, scale and pose. We set the minimum mask size η in eq. (3.3) to 25%, 20%, 15% and 15% for `car`, `horse`, `chair`, `bird` datasets respectively. In Figure 3.3 we show some examples of outputs produced by the generators from random



(a) Default parameters (b) No shift $\delta = 0$

(c) 25% shift $\delta = 0.25$ (d) Bg contrast jitter = (0.7, 1.3)



(e) No random crops (f) Mask size $\gamma_1 = 10.0$

(g) Min. mask size $\eta = 5\%$ (h) Single generator

Figure 3.4: Qualitative results of our approach for settings **(a)-(h)**: generated 64×64 composite images, masks, and outputs of Mask R-CNN.

samples in the Gaussian latent space. From the first to the fifth row in each quadrant: generated background layer, generated foreground layer, generated foreground mask layer, product between the mask and the foreground layer, and final composite image. As can be seen, the generator is able to learn very accurate object segmentations and texture. Also, it can be observed that the background has some residual artifacts in the center. This is due to the limited shift perturbation range, which does not allow the background layer to receive much feedback from the loss function during the training. In some cases the exact separation between object and background is not successful. This can be seen in the last two columns for each dataset.

3.4.1 Ablation study

To validate the design choices in our approach, we perform ablation experiments on the LSUN Car dataset. We introduce the following changes to **(a)** the default parameters described in Section 3.3, **(b)** disable the shift by setting the range of random location shift $\delta = 0$, **(c)** increase the shift to $\delta = 0.25 \times \texttt{resolution}$, **(d)** randomly jitter the background contrast in the range (0.7, 1.3) to further prevent the background from filling parts of objects, **(e)** directly resize real images to the desired resolution without random cropping, **(f)** increase the strength of the mask size loss $\mathcal{L}_{\text{size}}$ by setting its coefficient $\gamma_1 = 10$, **(g)** set the minimum mask size parameter to a smaller value $\eta = 5\%$, **(h)** use a single generator with 3 outputs for background, foreground and mask. To evaluate the quality of the generated segments, we generate 10,000 images and masks for each setting. We binarize our masks with a 0.5 threshold. To obtain an approximated mask ground truth on generated composite images we run Mask R-CNN [42, 83] pre-trained on MS-COCO [77] with a ResNet50 Feature Pyramid Network backend. If the car is detected, we evaluate the Intersection over Union (IoU) with the mask generated by our models on these images, defined as $\text{IoU} = \frac{|\text{m}_{\text{gen}} \cap \text{m}_{\text{pred}}|}{|\text{m}_{\text{gen}} \cup \text{m}_{\text{pred}}|}$. We run the evaluation on 64×64 and 128×128 pixels resolution, but resize the 64×64 images to 128×128 before feeding them to Mask R-CNN. The quantitative results can be found in Table 3.1 and the qualitative results in Figure 3.4.

Our ablation shows that the random shifts (see Section 3.2.2) are essential in our approach. When not used **(b)** the object segmentation fails and the objects are often in the background. The quality of the segmentation decreases drastically when the random shift range does not correspond to foreground object shifts in real images. This is illustrated by the setting **(c)** with large random shifts and by the setting **(e)**, where the real images are not randomly cropped. The additional random contrast jitter of the generated background **(d)** helps separate the foreground object from the background. A smaller value of the η parameter to ensure the minimum mask size (**g**) does not have a big impact on the results: It helps to avoid empty masks, but the mask size is mainly determined by the realism requirement. Using a single generator **(h)** to produce all

Table 3.2: FID scores comparison between our proposed GAN model and the single output (SO) GAN model.

Setting	**FID** (64×64)	**FID** (128×128)
SO GAN	27.807	21.665
Our GAN	31.409	30.867

outputs makes the background and the foreground too correlated, which prevents it from learning a good layered representation.

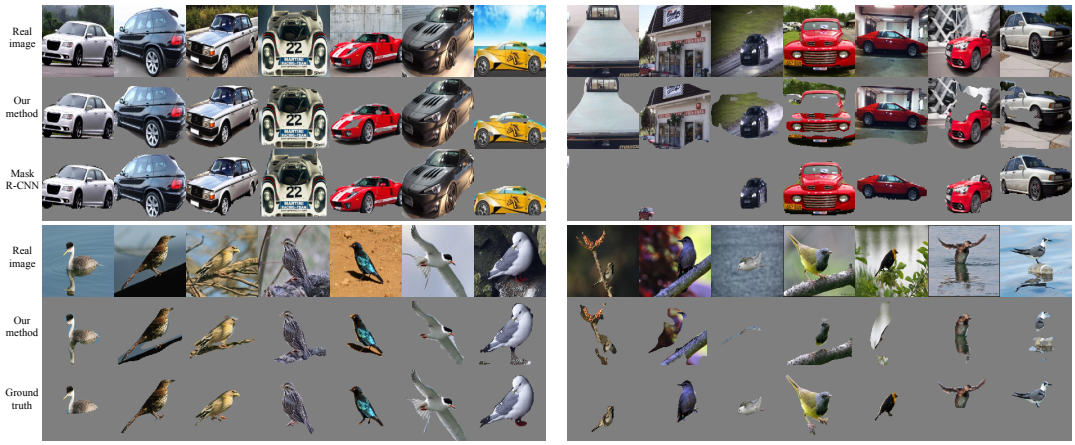
Quality of generated images. To evaluate the quality of generated composite images, we compute the Frèchet Inception Distance (FID [45]) using 10K real and 10K generated images composite images from our model (**d**). We compare it with a standard StyleGAN producing the entire images at once, trained for the same number of iterations. The results are presented in Table 3.2. The difference in the FID scores may be explained by the more demanding constraints of our model, which may hinder the GAN training.

Dataset with more than one object category. Although we run our experiments on datasets containing objects of one category, we argue that our method should work with multiple object categories when the GANs improve and are able to produce realistic images on diverse datasets. To verify this, we train our model on a dataset consisting of 50K images from LSUN Car and 50K images from LSUN Horse datasets. The qualitative results are presented in Figure 3.5. Although the quality of the generated images on such a dataset is lower, our model is still able to generate segmented scenes.

3.4.2 Segmenting real images

Finally, we train an encoder to find the segmentation of real images, as described in Section 3.2.3. We use our best generator trained on LSUN Car 64×64 images with background contrast jitter (setting **(d)**) and freeze its weights. We train an encoder that produces $5 \cdot 2 \cdot 2 = 20$ latent codes of 512 dimensions: For each of the 5 StyleGAN scales, we get 2 separate codes for AdaIN (Adaptive Instance Normalization) layers

Figure 3.5: Generated object segments using a dataset with two object categories: cars and horses.



(a) (b)

Figure 3.6: Qualitative results of segmentation on LSUN Car and CUB-200 datasets. (a) Examples of successful segmentations. (b) Examples of failures.

Table 3.3: Segmentation results comparison. For the LSUN Car dataset, Mask R-CNN generated segmentations from 10,000 images serve as ground truth. The IoU values are depicted for images with *detected* cars versus *all* images, including those using empty masks. CUB-200 comparisons employ real ground truth.

Setting	**LSUN Car** (detected)	**LSUN Car** (all)	**CUB-200** IoU
Our method	0.540	0.479	0.380
GrabCut	0.559	0.499	0.453
Full mask	0.402	0.357	0.132

in a convolutional block and get separate codes for 2 generators (background and foreground with mask). For the encoder, we use a randomly initialized ResNet18 network ([41]) with a 64×64 input without average pooling at the end and add a fully-connected layer with a $512 \cdot 20 = 10240$ output size. We feed the codes to the generator and minimize the autoencoder loss (eq. (3.7)). In the perceptual loss, we use our discriminator to extract $512 \times 8 \times 8$ spatial features on real and generated images. We evaluate the segmentation on the first 10,000 images of the LSUN Car dataset. We train separate encoders on chunks of 100 images as we found that it makes the encoding more stable than training on the entire dataset. We run the training for 1000 iterations with Adam optimizer, learning rate of 0.0001 and $\beta_1 = 0.9$, $\beta_2 = 0.999$. After training, we encode the images and feed the codes to the generator to obtain the masks. For the approximated ground truth we run Mask R-CNN on real images and

evaluate our segmentation with mean IoU. We also compute the IoU using the output of the GrabCut algorithm and a naive mask covering the entire image. The results are presented in Table 3.3. The performance of our method is capped by ambiguities in inverting the generator with an encoder, which is an active topic of research. We present sample segmentation results in Figure 3.6. We notice some failures, especially in the case of small objects. We repeat the same training and evaluation procedure on Caltech-UCSD Birds-200-2011 dataset [130], for which the segmentation ground truth is available. We use the parameters that worked best on the LSUN Car dataset for training both the generator and the encoders.

3.5 Discussion

We have introduced a new framework to learn object segmentation without using manual annotation. The method is based on the principle that valid object segments, when locally shifted relative to their background, can still yield a realistic image. The proposed solution is based on first training a generative network to learn an image decomposition model for a dataset of images and then on training an encoder network to assign a specific image decomposition model to each image. It strongly relies on the accuracy of the generative model, which today can be built with adversarial techniques. However, this framework is quite general and can be extended. For example, the current generative model postulates that a scene is composed only of a foreground and a background object, but an increase of output layers could allow describing scenes with multiple objects. In this case, the unknown number of objects and their interactions would need to be addressed.

Chapter 4

Unsupervised Learning of Object Segmentation From Perturbed Generative Models

In the previous chapter, we introduce a method that enables the generation of a layered image representation that separates the background from the masked foreground. The principle that shifting a segmented foreground object should yield a realistic scene encourages the generation of correctly segmented objects. However, while we can generate segmented objects, the fundamental problem we want to solve is object segmentation in real images. In this chapter, we introduce an approach to learn an object segmentation model directly from a large collection of images without any manual annotation. The key idea is to build a synthetic training set for segmentation (*i.e.*, where each sample consists of an input image and the corresponding segmentation mask) through a generative model. This dataset is then used to train a segmentation network in a supervised fashion. We explore and analyze a few different methods, including a single end-to-end training of both the generative model and the segmenter that can be applied directly to real images. We demonstrate on several datasets that models trained on the generated data are able to generalize well on real images. Most notably, we explore training our model on a small, but diverse dataset of images. We show that we are able to learn segmentation even when the quality of the generated images is subpar due to GAN limitations on small diverse datasets. So the object segmentation can still be accurate although the image generation may not be extremely realistic. Furthermore, we update and simplify our approach by using a single conditional generator for foreground and background, which reduces the model complexity and size compared to the previous approach.

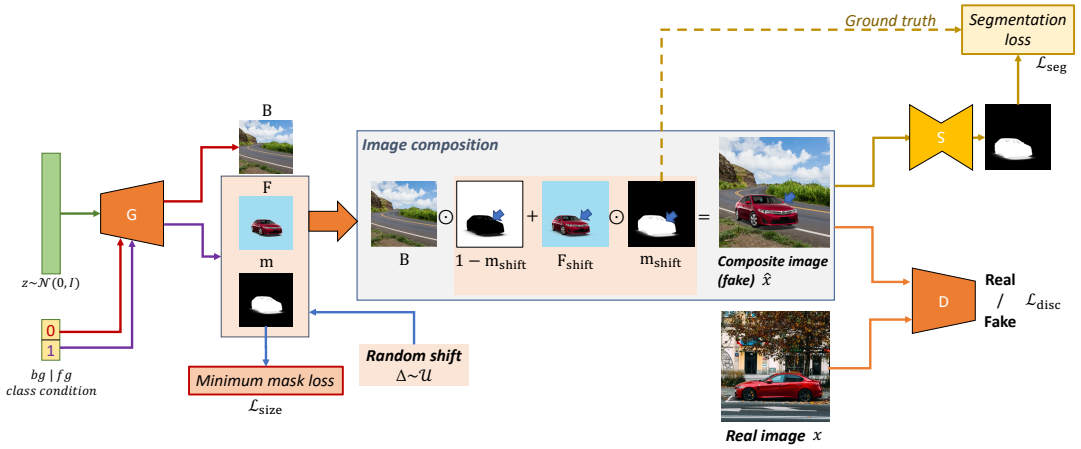


Figure 4.1: Illustration of the proposed architecture to learn to generate realistic layered scene representations through G and to learn to segment real images through the segmentation network S trained on the generated composite images and masks. The layered representation consists of three components: 1) a background image, 2) a foreground image, and 3) a(n alpha matte) mask image. The generator is conditioned on the background/foreground class and is run twice with the same noise z for both classes to obtain all the components of the representation. A crucial component of our model is the generation of random shifts Δ of the foreground object (in particular, such that they are independent of the input vector z to G) during the training of the generator. The generator is trained adversarially against a discriminator D. The segmentation network S is trained either with a pre-trained generator or jointly as part of the U-Net discriminator (not shown in the illustration).

With reference to Figure 4.1, we propose to build a generative model that outputs a background image, a foreground object, and a foreground mask, similarly to the method of Chapter 3. This model is trained in an adversarial manner against a discriminator. The discriminator aims to distinguish the composite image, which is obtained by overlaying the generated background and the shifted foreground and mask, from real images. Then, to learn to segment an image, we propose to train a segmentation network with the generated layered representation. In other words, given a triplet background image, foreground image, and foreground mask, the segmentation network would be trained to map the composite image to its corresponding foreground mask. The segmentation network can be trained either jointly end-to-end with the generative model as part of the discriminator or separately after the generative model is trained. We demonstrate that, provided that the generator is trained well, the segmentation network trained with generated data also generalizes to real images. Finally, these two steps define our fully unsupervised learning approach to segment objects. We do not make

use of object detectors, classifiers, bounding boxes, landmarks, or pre-trained networks. Moreover, the proposed method is quite general, as we demonstrate qualitatively on several object categories, and quantitatively on the LSUN Car [143], CUB-200-2011 [130], Flowers102 [93] datasets as well as a small diverse saliency detection dataset DUTS [131], containing multiple categories of objects.

Contributions. This work introduces a new approach to learning unsupervised object segmentation models using perturbed generative models. We explore various strategies for training a segmenter with generated layered representations, including end-to-end training with the generative model itself. Our method proves to generalize effectively to real-world images, even when the quality of generated data is subpar due to the inherent limitations of GANs on small and diverse datasets. We evaluate our models, achieving competitive results on several benchmarks. Additionally, we refine the generative model's architecture by integrating a conditional generator, which cuts the number of model parameters by half.

4.1 Learning to Segment without Supervision

Our approach is based on training two main models: A generator G and a segmentation network S (see Figure 4.1 for an illustration of the proposed method). In the first model, the generator is trained against a discriminator in an adversarial manner with a modern Generative Adversarial Network [57]. The generator G learns to generate composite scene samples in the form of a foreground and a background layer, to the extent that the discriminator cannot distinguish them from real images. In the second model, we train a segmentation network S to segment images. We make use of the generator outputs to produce composite images and corresponding objects masks pairs. We then use these pairs to train a model that maps the composite images to the masks. We find that if the generated distribution is close enough to that of real images, the segmentation network is able to produce meaningful segmentations for real images as well. In the following sections, we explain our approach more in detail.

4.1.1 A Generative Model of Layered Scenes through Model Perturbation

Similarly to Section 3.2.1, we consider an $M \times N$ discrete image domain $\Omega \subset \mathbb{Z}^2$. We define the representation of a scene as a layered composition of a background image $\mathrm{B} : \Omega \mapsto \mathbb{R}$, a foreground image $\mathrm{F} : \Omega \mapsto \mathbb{R}$ with a foreground mask $\mathrm{m} : \Omega \mapsto [0, 1]$, and the composite image $\bar{x} : \Omega \mapsto \mathbb{R}$ defined at each pixel $\mathbf{p} \in \Omega$ as

$$\bar{x}[\mathbf{p}] = (1 - \mathrm{m}[\mathbf{p}])\mathrm{B}[\mathbf{p}] + \mathrm{m}[\mathbf{p}]\mathrm{F}[\mathbf{p}]. \tag{4.1}$$

This time we define a conditional generator $\mathrm{G} : \mathbb{R}^k \times \{0,1\} \mapsto \mathbb{R}^{2\times M\times N}$ through a Convolutional Neural Network such that, given a k-dimensional input vector $\mathrm{z} \sim \mathcal{N}(0, I_d)$, it outputs two components for the background class 0, *i.e.*, $\mathrm{G}(\mathrm{z}, 0) = [\mathrm{G_B}(\mathrm{z}), \mathrm{G_{mbg}}(\mathrm{z})]$ and two for the foreground class 1, *i.e.*, $\mathrm{G}(\mathrm{z}, 1) = [\mathrm{G_F}(\mathrm{z}), \mathrm{G_m}(\mathrm{z})]$. Then, we let $\mathrm{B} \doteq \mathrm{G_B}(\mathrm{z})$, $\mathrm{F} \doteq \mathrm{G_F}(\mathrm{z})$ and $\mathrm{m} \doteq \mathrm{G_m}(\mathrm{z})$. We discard the background mask output $\mathrm{G_{mbg}}(\mathrm{z})$. The generator is then trained in an adversarial manner against a discriminator neural network $\mathrm{D} : \mathbb{R}^{M\times N} \mapsto \mathbb{R}$ (see Figure 4.1). Our implementation is based on StyleGAN2 [57].

Similarly to Section 3.2.2, we define the composite image with the foreground object translated by a small shift $\Delta \sim \mathcal{U}([-\delta, \delta] \times [-\delta, \delta])$ as

$$\hat{x}[\mathbf{p}] = (1 - \mathrm{m}[\mathbf{p} + \Delta])\mathrm{B}[\mathbf{p}] + \mathrm{m}[\mathbf{p} + \Delta]\mathrm{F}[\mathbf{p} + \Delta]. \tag{4.2}$$

as well as loss terms ensuring that the mask is not empty

$$\mathcal{L}_{\text{size}} = \mathbb{E}_{\mathrm{z}\sim\mathcal{N}(0,I_d)}\left[\max\left\{0, \eta - \frac{1}{MN}|\mathrm{G_m}(z)|_1\right\}\right] \tag{4.3}$$

with a mask size parameter $\eta \in (0, 1)$, and a loss that encourages the binarization of the mask

$$\mathcal{L}_{\text{binary}} = \mathbb{E}_{\mathrm{z}\sim\mathcal{N}(0,I_d)}\big[\min\{\mathrm{G_m}(z), 1 - \mathrm{G_m}(z)\}\big]. \tag{4.4}$$

To train the generator, we minimize the updated StyleGAN2 loss with respect to $\mathrm{G_B}$, $\mathrm{G_F}$ and $\mathrm{G_m}$

$$\mathcal{L}_{\text{gen}} = \mathbb{E}_{\hat{x}\sim p_{\hat{x}}}[\log(1 - \mathrm{D}(\hat{x}))] + \gamma_1\mathcal{L}_{\text{size}} + \gamma_2\mathcal{L}_{\text{binary}} \tag{4.5}$$

with $\gamma_1, \gamma_2 > 0$ and where $p_{\hat{x}}$ is the probability density function of $\hat{x}$ as defined via eq. (4.2) with $z \sim \mathcal{N}(0, I_d)$ and $\Delta \sim \mathcal{U}([-\delta, \delta] \times [-\delta, \delta])$. To train the discriminator we minimize the following loss with respect to D

$$\begin{aligned}\mathcal{L}_{\text{disc}} =& \mathbb{E}_{\hat{x}\sim p_{\hat{x}}}[\log(1 + \mathrm{D}(\hat{x}))] + \mathbb{E}_{x\sim p_x}[\log(1 - \mathrm{D}(x))] \\ &+ \lambda\mathbb{E}_{x\sim p_x}\left[|\nabla_x \mathrm{D}(x)|_2^2\right]\end{aligned} \tag{4.6}$$

where p_x is the probability density function of real images x and $\lambda > 0$ is the R1 regularization strength, as described in StyleGAN2 [57].

4.1.2 Object Segmentation Trained on Generated Data

For the generated composite image $\hat{x}$ the object mask $\mathrm{G_m}(\mathrm{z})$ is one of the generator outputs and is known (see Figure 4.1). Therefore we can train a foreground-background segmentation network $S : \mathbb{R}^{M\times N} \mapsto \mathbb{R}^{M\times N}$ that maps an image $\hat{x}_\Delta$ to the corresponding

mask $\mathrm{G_m(z)}$. To train the segmentation network, we optimize the binary cross-entropy loss

$$\mathcal{L}_{\text{seg}} = \mathbb{E}_{z\sim\mathcal{N}(0,I_d)}\left[\frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N} -\mathrm{G}_{\mathrm{m}}^{i,j}(\mathrm{z})\log(S^{i,j}(\hat{x}))\right], \tag{4.7}$$

where we have used the $A^{i,j}$ notation to indicate the (i,j)-th entry of the 2D array A.

We choose the U-Net [107] architecture for the segmentation network (briefly described in Section 2.2.2, page 24) and explore three ways to train it:

1. **U-Net discriminator with mask prediction.** We use a U-Net discriminator [111] and train the segmentation network as part of the discriminator. The U-Net discriminator outputs $\mathrm{D}(x)$ as the global encoder output as well as $\mathrm{D_{dec}}(x) \in \mathbb{R}^{M\times N}$ as the *decoder* output, which is a pixel-specific measure of image realism. The adversarial loss term for this pixel output is

$$\begin{aligned}\mathcal{L}_{\text{disc\_dec}} = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\Bigg(&- \mathbb{E}_{\hat{x}\sim p_{\hat{x}}}\left[\log(1-\mathrm{D}_{\mathrm{dec}}^{i,j}(\hat{x}))\right] \\ &- \mathbb{E}_{x\sim p_x}\left[\log(\mathrm{D}_{\mathrm{dec}}^{i,j}(x))\right] \\ &+ \lambda_{\mathrm{dec}}\mathbb{E}_{x\sim p_x}\left|\nabla_x \mathrm{D}_{\mathrm{dec}}^{i,j}(x)\right|_2^2\Bigg).\end{aligned} \tag{4.8}$$

   We add another output channel $\mathrm{D_{seg}}$ to the U-Net decoder so that $\mathrm{S(x)} \doteq \mathrm{D_{seg}(x)}$ represents the predicted object mask. We optimize jointly the segmentation and discriminator losses

$$\mathcal{L}_{\text{unetdisc\_seg}} = \mathcal{L}_{\text{disc}} + \mathcal{L}_{\text{disc\_dec}} + \mathcal{L}_{\text{seg}}. \tag{4.9}$$

2. **Discriminator with U-Net mask prediction.** We can also omit the U-Net discriminator output and use the decoder part only for the mask prediction. This can be achieved by optimizing

$$\mathcal{L}_{\text{disc\_seg}} = \mathcal{L}_{\text{disc}} + \mathcal{L}_{\text{seg}}. \tag{4.10}$$

3. **U-Net post-training with generated data.** We use the pre-trained generator to create the training data: pairs of the composite image and corresponding mask. We use the composite image as the input and the mask as pseudo-labels to train the segmentation network. The generator can be trained with or without the mask prediction losses from previous methods.

Table 4.1: Comparison of different segmentation methods on the DUTS-TR and ECSSD saliency datasets.

Setting	**Joint training IoU**		**Post-training IoU**		**FID ↓**
	DUTS-TR	ECSSD	DUTS-TR	ECSSD	
(a) U-Net discriminator + mask	0.687	0.595	0.727	0.659	52.11
(b) U-Net discriminator + mask + cutmix	0.681	0.609	0.736	0.658	46.01
(c) Standard discriminator + U-Net mask prediction	0.726	0.656	0.719	0.651	52.87
(d) Standard discriminator → post-training only	-	-	0.739	0.653	50.27

4.2 Implementation

In all experiments, the network architectures and details follow StyleGAN2 [57], except where noted otherwise in this section. We use the conditional variant of StyleGAN2 [56] and modify it to output a 3 color channel image and a single channel mask. We sample the generator with the same 512 dimensional Gaussian latent code, but with different conditional labels to get the outputs that we use for the background (label 0) and the foreground with the mask (label 1). We ignore the mask output for the background and apply a sigmoid activation function to the foreground mask. This reduces the generator model size by half compared to our previous approach, where two separate generators are used for background and foreground generation.

We use mixing with probability 0.9 and feed two latent codes to two parts of the generator split by a randomly selected crossover point. We set the input and generated image resolution `resolution` to 64 or 128 and use a batch size of 64. The local shift range δ described in Section 4.1.1 is set to $\delta = 0.125 \times$ `resolution`. We randomly jitter the background and foreground contrasts in the range (0.7, 1.3) to further prevent the background from filling parts of the foreground objects. To match the shifts in the generated data, we first resize the center crop of the real image to a square image of size $1.125 \times$ `resolution` and then take a random crop of size `resolution`. We train the StyleGAN2 network on real images x and composite images $\hat{x}$ (eq. (4.2)) by alternatively minimizing the discriminator loss (eq. (4.8)) and the generator loss (eq. (4.5)). We set the R1 regularization strength in the discriminator loss to $\lambda = 0.1$. For the U-Net discriminator, we add the decoder by mirroring the encoder and use transposed convolutions for upsampling. We then adapt the number of channels in each layer so that the number of learnable parameters is roughly the same as that of the single binary output discriminator. We regularize the U-Net discriminator with

cutmix consistency [111]. In the generator loss we set $\gamma_1 = 5$ for the minimum mask size term and $\gamma_2 = 2$ for the binarization term. We optimize our GAN with the Adam optimizer [61] and parameters $\beta_1 = 0$, $\beta_2 = 0.99$. We use a fixed learning rate of 0.001.

To evaluate our U-Net mask predictor, we first resize the input image to keep the same scale as during the training, i.e., so that the smaller side is $\delta = 0.125 \times$ `resolution`. Since U-Net is fully convolutional, we can run a forward pass on these bigger images, occasionally adding reflective padding when the size of the feature maps from concatenated skip connections does not match.

4.3 Experiments

4.3.1 Datasets

We experiment with 4 datasets of different size and variety. DUTS [131], in particular, includes multiple categories, presenting a challenge for current GAN models, which tend to generate higher quality images with large, single-category datasets when additional labels are not utilized.

LSUN Car [143]. This dataset exhibits a wide range in position, scale, and pose. We utilize the first 100,000 images of the dataset for training and another 10,000 for evaluation. As the dataset lacks segmentation annotations, we derive an approximate mask ground truth for evaluation on generated composite images by employing Mask R-CNN [42], [83] pre-trained on MS-COCO [77] with a ResNet50 Feature Pyramid Network backend.

Caltech-UCSD Birds-200-2011 (CUB-200-2011) [130]. We use the data split from [18]: We train on $10K$ images, then use $1K$ images for the test split and 788 for validation.

Flowers102 [93]. The dataset consists of 8,189 images of flowers and masks obtained with an automated method specific to this dataset. We use a data split from [18]: 6,149 training images, 1,020 validation images and 1,020 test images.

DUTS [131]. The DUTS dataset stands as a significant challenge for current generative and segmentation models, containing a wide array of objects drawn from diverse ImageNet and SUN categories. It contains 10,553 training images and 5,019 test images.

4.3.2 Ablation study

We compare the segmentation methods described in Section 4.1.2 to determine the most effective approach for using our layered generative model to segment real images. We use the default parameters from Section 4.2 but set the minimum mask size to

$\eta = 0.15$ and train the models on the training split of the DUTS dataset. We explore the joint generator and segmenter training in the following settings:

(a) A U-Net discriminator with mask prediction (eq. (4.9))

(b) Same as **(a)** with cutmix regularization [111]

(c) A standard discriminator with U-Net decoder used only for mask prediction (eq. (4.10))

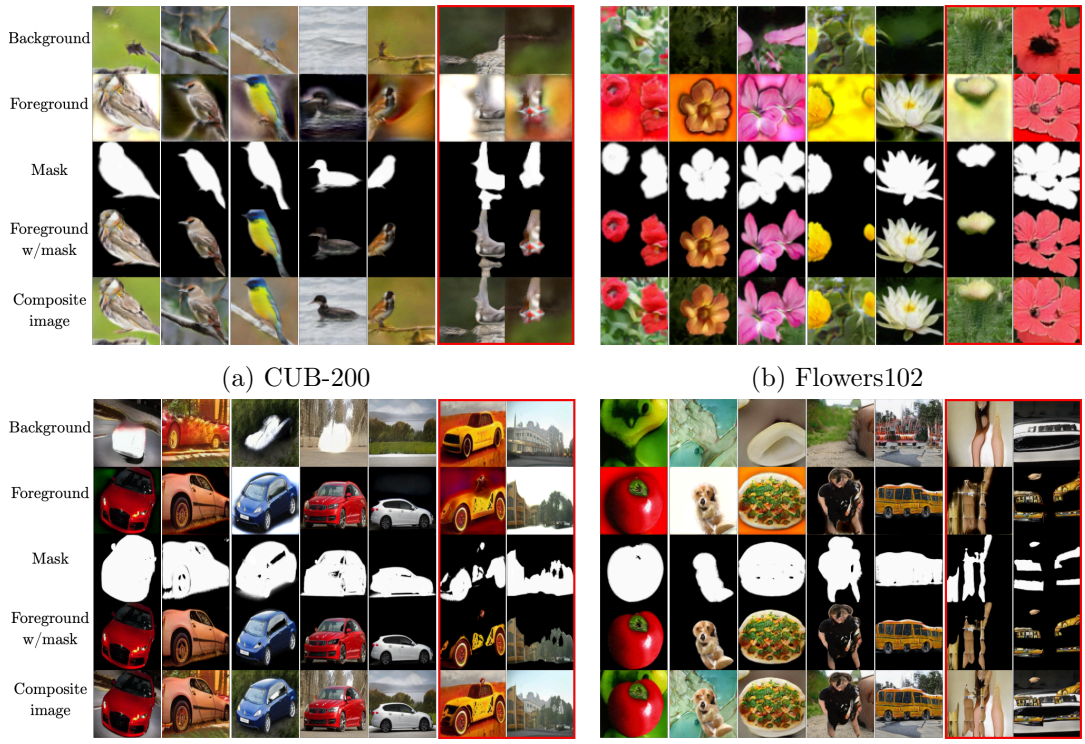
On top of that, we explore the two-step training **(d)** when we first train just the generative model with a standard discriminator and then use the pre-trained generator to synthesize composite images and pseudo ground-truth to train a U-Net segmenter. Note, that we can apply the same approach to settings **(a)-(c)**, by discarding the segmenter trained jointly with the generator and training a new one utilizing only a pre-trained generative model. In this case, we train the segmentation network for $150K$ iterations with a batch size of 64 and a learning rate of 0.001, which we decrease to 0.0002 after $75K$ iterations.

We use the ECSSD saliency detection dataset [113] of $1K$ images as a validation set and report the IoU with the corresponding saliency ground truth. We also show the evaluation on the training set as it can be useful in the unsupervised setting. To get a measure of the quality of the generated composite images, we compute the FID [45] using statistics from $5K$ images from the training set.

Quantitative results can be found in Table 4.1. We notice that training the U-Net discriminator jointly with its mask prediction head (settings **(a)**, **(b)**) gives worse segmentation results than training a standard discriminator (*i.e.*, just the encoder part) with the decoder focused solely on the mask prediction **(c)**. In most of the settings, using the pre-trained generative model to generate synthetic data to then train a U-Net segmentation network shows improved results that are similar across the settings. Only in setting **(c)**, the segmentation model trained jointly with the generative model which is on-par with post-training, proving to be a good choice for a single end-to-end training. We get the best quality of generated images when using a U-Net discriminator with cutmix consistency regularization, as shown by the lower FID value for this setting. For this reason, for the remainder of the experiments we choose to train a generative model with setting **(b)** and then use it to generate data to separately train a segmentation model.

4.3.3 Segmentation results

We train our model on LSUN Car, Flowers102, CUB-200-2011, and DUTS datasets and compare our results to other methods. For the CUB-200 and Flowers102 datasets,



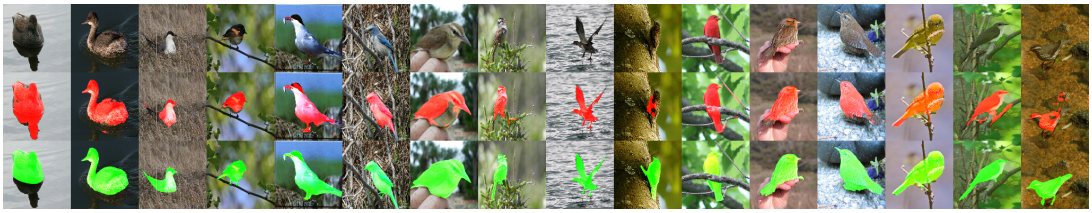
(a) CUB-200 (b) Flowers102

(c) LSUN Car (d) DUTS

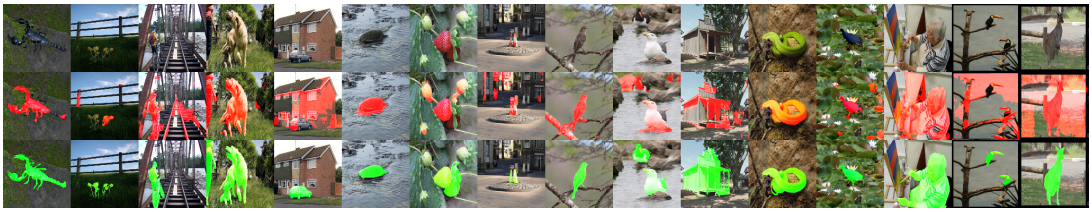
Figure 4.2: Generated backgrounds, foregrounds, masks, foregrounds with mask applied and composite images for 4 different image datasets. Last two columns in each dataset show generator failures, *e.g.*, an object in the background or an unrealistic foreground.

we set the generator's resolution to 64×64; for the DUTS and LSUN Car datasets, we increase the resolution to 128×128. Except for LSUN Car, we train the generative model for $1K$ epochs, which corresponds to $150K$ iterations on DUTS and CUB-200 datasets and $90K$ iterations for Flowers102. For the larger LSUN Car dataset we train for $600K$ iterations. For CUB-200 and Flowers102 we adopt the evaluation strategy of other methods and compute the metrics on center crops of the images. We report intersection over union between the predicted and ground truth masks, $\text{IoU} = \frac{|\text{m}_{\text{pred}} \cap \text{m}_{\text{gt}}|}{|\text{m}_{\text{pred}} \cup \text{m}_{\text{gt}}|}$, pixel accuracy, and $F_\beta = \frac{(1+\beta^2)\text{Precision} \times \text{Recall}}{\beta^2 \text{Precision} + \text{Recall}}$ score with $\beta^2 = 0.3$. Following previous works [85], we report F_β for the saliency dataset and $\max F_\beta$[1] for Flowers and CUB-200 datasets.

[1] $\max F_\beta$ is the maximum F_β value selected from different binarization thresholds

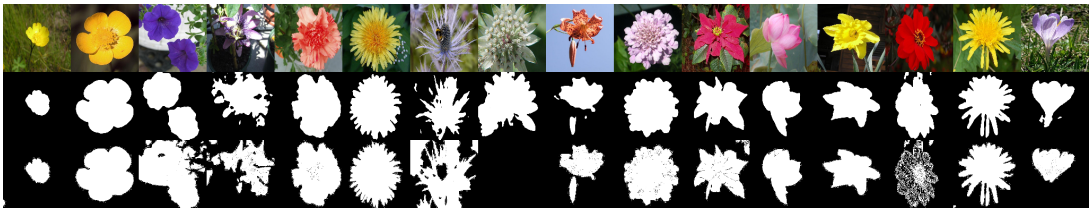


(a) CUB-Birds



(b) DUTS-TE

(c) LSUN Car



(d) Flowers102

Figure 4.3: Qualitative (not cherry-picked) results of segmentation on CUB-200, DUTS, LSUN Car and Flowers102 datasets. **First row:** segmented image. **Second row:** segmentation results. **Third row:** ground truth segmentation.

Qualitative results of the generative model

In Figure 4.2 we show some examples of outputs produced by the generators from random samples in the Gaussian latent space. From the first to the fifth row in each quadrant: generated background layer, generated foreground layer, generated foreground mask, product between the mask and the foreground layer, and final composite image. As shown, the generator is able to learn very accurate foreground-background segmentations and texture. In some cases, the exact separation between object and background is not successful. This can be seen in the last two columns for

each dataset. Simpler datasets containing one category of images, like CUB-200 and Flowers102, show especially good results. DUTS is a relatively small dataset with a large variety of objects, a setting that is challenging for current GAN methods. While the quality of the generated images is worse than in the case of the other datasets, we can still observe some meaningful foreground objects being generated. LSUN Car is a larger dataset that does not always have a main object of interest, has a lot of outliers, and shows a large variety in appearance, scale, pose, and position of the objects. We are able to generate meaningful segmentations and textures of cars, but we also notice failure cases more often.

Qualitative results of the segmentation model

We show qualitative results of our segmentation prediction for each dataset in Figure 4.3, where the first row is a real image from the test set, the second row is our mask prediction and the third row is the ground truth segmentation (or approximated ground truth for LSUN Car). We can see how some discrepancies between our masks and the ground truth may result from the learning rule that we employ: That is, in the first image of the CUB-Birds dataset, our model selects the bird with its reflection on the water, since probably it has learned that shifting the bird alone would not render a realistic image. We find that we are often able to produce meaningful segmentations for the DUTS dataset, although the quality of the generated images for DUTS is somewhat lacking due to the small size of the dataset, the presence of multiple classes, and current GAN limitations. There is also an inherent ambiguity in the choice of what object to segment. Thus, our evaluation with a fixed ground truth provides a limited view on the actual performance of our trained segmentation network. Segmentations for the LSUN car dataset are sometimes better than the approximated ground truth that we obtained with the supervised segmentation model [83]. The ground truth masks for Flowers102 were also generated with automated methods and in many cases our masks seem to be more precise.

Quantitative comparison with other methods

We compare our method with several other approaches for unsupervised segmentation. The results are in Tables 4.2, 4.3, 4.4, and 4.5 for CUB-200, Flowers102, LSUN car, and DUTS respectively. ReDO [18] and IEM [110] are two methods designed for unsupervised segmentation, one relying on training a GAN to redraw object segments, the other on maximizing the inpainting error over the two partitions of the image. Voynov [129] and Melas-Kyriazi [85] find a direction in the ImageNet pre-trained GAN's latent space that can be used for background removal and then train a segmentation network with the generated data. HS [138], wCtr [156] and WSC [73] are unsupervised

Table 4.2: Comparison of unsupervised segmentation results on the CUB-200-2011 test set. Methods with ⋆ use extra data (e.g. ImageNet for unsupervised GAN training)

Method	**IoU**	**Accuracy**	**max**F_β
Voynov⋆ [129]	0.683	93.0	0.794
Voynov-E⋆ [129]	0.710	94.0	0.834
Melas-Kyriazi⋆ [85]	0.664	92.1	0.783
PerturbGAN [10]	0.360	-	-
ReDO [18]	0.426	84.5	-
IEM [110]	0.551	89.3	-
Ours	**0.784**	**96.1**	**0.890**

Table 4.3: Comparison of unsupervised segmentation results on the Flowers102 dataset. Methods with ⋆ use extra data (e.g. ImageNet for unsupervised GAN training)

Method	**IoU**	**Accuracy**	**max**F_β
Voynov⋆ [129]	0.540	76.5	0.760
Voynov-E⋆ [129]	0.804	90.4	0.878
Melas-Kyriazi⋆ [85]	0.541	79.6	0.723
ReDO [18]	0.764	87.9	-
IEM [110]	0.789	89.6	-
Ours	**0.807**	**90.4**	**0.884**

Table 4.4: Comparison of unsupervised segmentation results on the LSUN car dataset. ⋆There are some differences between test sets in each method (that is why we do not indicate the top performance in boldface).

Method	**IoU**	**Accuracy**
PerturbGAN⋆ [10]	0.54	-
IEM⋆ [110]	0.632	77.8
Ours	0.621	84.8

methods for saliency detection. We cannot directly compare the metrics on the LSUN Car dataset since there are no published data splits for this dataset; we use $10K$ images that we did not use in the training sets. In the Flowers102 dataset, the ground truth is computed automatically from the images and its accuracy is not always high (*e.g.*, we see in Figure 3.6 several cases where the ground truth mask is empty or inverted, while our method provides a meaningful segmentation). For all datasets, we show that

Table 4.5: Comparison of saliency detection methods on the DUTS dataset. † results from [85]. ‡ results from [92]. ⋆ methods use extra data (e.g. ImageNet for unsupervised GAN training). ⋆⋆ initialized with a pre-trained supervised segmentation network.

Method	**IoU**	**Accuracy**	F_β
Handcrafted Methods			
RBD‡ [156]	-	-	0.510
DSR‡ [74]	-	-	0.558
MC‡ [51]	-	-	0.529
HS‡ [138]	-	-	0.521
Deep Ensembles of Handcrafted Methods			
SBF‡ [146]	-	-	0.583
USD⋆⋆‡ [150]	-	-	0.716
USPS⋆⋆ [92]	-	-	0.736
Unsupervised Methods			
Voynov⋆† [129]	0.508	88.1	0.600
Melas-Kyriazi⋆ [85]	**0.528**	**89.3**	**0.614**
Ours	0.517	88.6	0.613

we outperform or match other unsupervised methods in terms of accuracy, IoU, and F_β without relying on any models pre-trained on bigger datasets.

4.4 Discussion

We have introduced a new framework to learn foreground-background segmentation without using manual annotation. The key idea is to use generated images of scenes and corresponding object masks as pseudo ground-truth, which can be used to train a segmentation network in a supervised manner. This can be done jointly with training the layered generative model, which allows for a single end-to-end training of the segmenter from a collection of images, or by training the segmenter separately while using a pre-trained generator to produce a synthetic dataset. To build such a generative model in a completely unsupervised way we propose to train a *conditional* generator by using the principle that valid object segments can be locally shifted relative to their background and still yield a realistic image. As we show in our experiments, this principle allows a generative network to learn an image decomposition model from a dataset of images. The quality of the generated dataset strongly relies on the accuracy of the generative model, which we built with modern adversarial techniques.

However, we show that even though the quality of generated composite images may be subpar due to GAN limitations, we are still able to train accurate segmentation models that generalize to real-world datasets. We expect that further progress in generative modeling will improve both the layered image generation and the generalization of segmenters trained on such synthetic datasets.

Chapter 5

MOVE: Unsupervised Movable Object Segmentation and Detection

In Chapters 3 and 4 we introduce methods that utilize Generative Adversarial Networks and the cohesive motion property of objects to train unsupervised object segmentation models from a collection of images only. Our learning rule requires shifting the predicted object in a scene. By using a generative model to produce separate background and foreground layers, which would later be composed into a scene, we avoid the necessity to inpaint the background behind the object, which would otherwise be exposed if we were to shift objects in a real image. However, relying on generating scenes with GANs has its limitations, which become apparent in the experiments. The quality of generated data depends on the size of the dataset, homogeneity, and objects' alignment. The variety of outputs may be limited due to common problems with mode collapse.

We now introduce MOVE, a novel method to segment objects without any form of supervision. As a proxy signal, we use the *movability* of objects, *i.e.*, whether they can be locally shifted in a realistic manner. This property holds for objects in the foreground, as they occlude all other objects in the scene. This basic idea has already been exploited in prior work with relative success. Nevertheless, here we introduce a novel formulation based on movability that yields a significant performance boost across several datasets for salient object detection.

In our approach, it is not necessary to move objects far from their initial location or to other images [3, 97] and thus we do not have to handle the context mismatch. It is also not necessary to employ models to generate entire scenes [10, 140], which can be challenging to train. Instead, in MOVE we rely on inpainting and show how we can use it to our advantage to get an additional signal driving object segmentation.

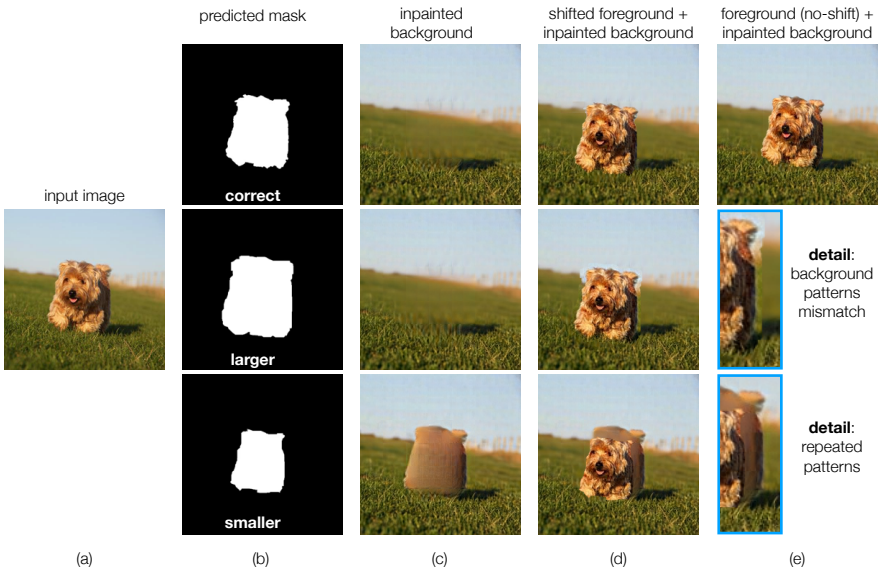


Figure 5.1: Exploiting inpainting and movability. (a) Input image. (b) Examples of predicted segmentation masks: correct (top), larger (middle), and smaller (bottom). (c) Inpainted backgrounds in the three corresponding cases. (d) Composite image obtained by shifting the foreground object in the three cases. (e) It can be observed that when the mask is incorrect (it includes parts of the background or it does not include all of the background), the background inpainting combined with shifting reveals repeated patterns and mismatching background texture when compared to the original input image or composite images obtained without shifting.

Suppose that, given a single image (Figure 5.1 (a)), we predict a segmentation mask (one of the 3 cases in Figure 5.1 (b)). With the mask, we can remove the object and inpaint the background (Figure 5.1 (c)). Then, we can also extract the foreground object, randomly shift it locally, and paste it on top of the inpainted background (Figure 5.1 (d)). When the mask does not accurately follow the outline of a foreground object (e.g., as in the middle and bottom rows in Figure 5.1), we can see duplication artifacts (of the foreground or of the background). We exploit these artifacts as a supervision signal to detect the correct segmentation mask. As the inpainter, we use a publicly available Masked AutoEncoder (MAE) [44] trained with an adversarial loss.[1] Inpainting may also introduce artifacts unrelated to the incorrect segmentation mask, which cannot be fixed and may affect the detection of the artifacts we are concerned with. However, we propose methods to minimize their impact. Our segmenter uses a pre-trained SSL ViT as a backbone (e.g., DINO [17] or the MAE encoder [44]). We then train a neural network head based on an upsampling Convolutional Neural

[1] https://github.com/facebookresearch/mae/blob/main/demo/mae_visualize.ipynb

Network (CNN). Following [114], we also further refine the segmenter by training a second segmentation network (SelfMask [114]) with supervision from pseudo-masks generated by our trained segmenter. Even without these further refinements MOVE shows a remarkable performance on a wide range of datasets and tasks. In particular, in unsupervised single object discovery on VOC07, VOC12 and COCO20K it improves the SotA CorLoc between 6.1% and 9.3%, and in unsupervised class agnostic object detection on COCOval2017 it improves the AP_{50} by 6.8% (a relative improvement of 56%), the AP_{75} by 2.3% (relative 55%) and the AP by 2.7% (relative 49%).

5.1 Background

In Section 2.3.4, page 28, we presented an overview of self-supervised methods and how they can be used for unsupervised object segmentation. Most prior work based on SSL features defines some form of clustering by either using attention maps [2, 135, 142] or similarity graphs [114, 115, 136]. Extracting masks directly from features usually produces coarse masks due to downsampling in Convolutional Neural Networks or tokenization of image patches in Vision Transformers. To improve the precision of the generated masks, either strong post-processing [7, 136] or smoothing via post-training [84, 114, 135] is required. In contrast, MOVE learns to produce high-resolution precise masks directly from images via our *movability* training rule. We use self-supervised models as a strong backbone and as an inpainter, instead of developing methods that extract masks from the features directly for each image. Our working principle partly exploits observations also made by [58, 110, 141]. They point out that the correct mask maximizes the inpainting error both for the background and the foreground. However, using the inpainting reconstruction error as a supervision signal may be too ambiguous to segment the entire objects precisely. Instead, we rely on the detection of artifacts generated through shifting, which we find to provide stronger guidance.

5.2 Method

Our objective is to train a segmenter to map a real image $x \in \mathbb{R}^{H \times W \times 3}$, with H the height and W the width of the image, to a mask $m \in \mathbb{R}^{H \times W}$ of the foreground, such that we can synthesize a realistic image for any small shifts of the foreground. The mask allows to cut out the foreground from x and to move it arbitrarily by some $\delta \in \mathbb{R}^2$ shift (see Figure 5.2, top-left). However, when the shifted foreground is copied back onto the background, missing pixels remain exposed. Thus, we inpaint the background

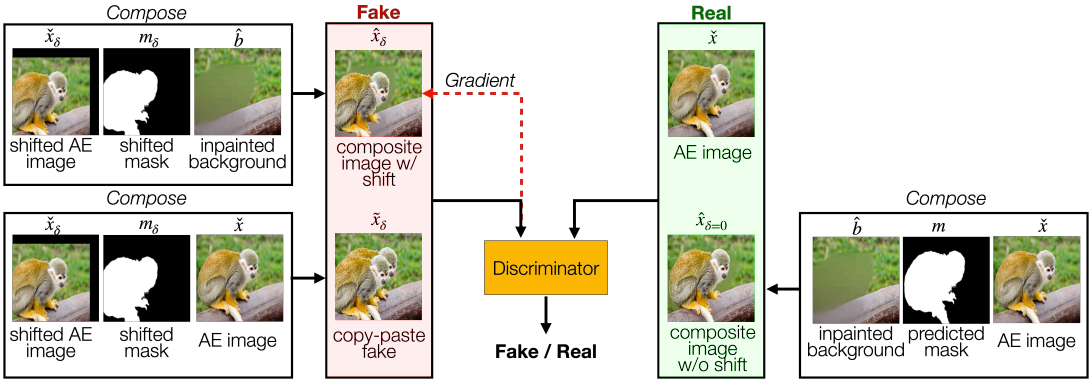


Figure 5.2: Synthetic and real images used to learn how to segment foreground objects. We obtain the predicted mask and inpainted background from our segmenter and MAE respectively. We train the segmenter in an adversarial manner so that the composite image with a shifted foreground (left, top row) looks real. A discriminator is trained to distinguish two types of real (right) from two types of fake (left) images. The fake images consist of the composite image with a shift and a copy-paste image, obtained by placing the shifted foreground on top of the input image. The set of real images consists of composite images without a shift and the real images. The real images are first autoencoded with MAE to match the artifacts of the inpainted background.

with a *frozen* pre-trained MAE[2] and obtain $\hat{b}$ (see Figure 5.3). Moreover, there is a difference between the texture of $\hat{b}$, which is generated from a neural network, and the texture of the cut out foreground from x, which is a real image. To ensure more similarity between these two textures, we synthesize $\hat{x}_\delta$ by extracting the foreground from the autoencoding (AE) of the input image x shifted by δ, which we call $\check{x}_\delta$, and by pasting it onto the background $\hat{b}$.

We enforce the realism of the synthesized images $\hat{x}_\delta$ by using adversarial training, *i.e.*, by training the segmenter against a discriminator that distinguishes two sets of *real* (Figure 5.2, right hand side) from two sets of *fake* images (Figure 5.2 left hand side). The synthetic *real* image $\hat{x}_{\delta=0}$ is obtained by composing a zero-shifted foreground with the inpainted background; the second *real* image $\check{x}$ is instead simply the AE of x. The two *fake* images are obtained by composing a δ-shifted foreground with either the inpainted background $\hat{b}$ or $\check{x}$, and obtain $\hat{x}_\delta$ and $\tilde{x}_\delta$ respectively.

We introduce all the above synthetic images so that the discriminator pays attention only to artifacts due to incorrect masks from the segmenter. Ideally, the segmenter should generate masks such that the fake image $\hat{x}_\delta$ looks as realistic as $\check{x}$ for any small δ. However, the discriminator might distinguish these two images because of the

[2]The MAE [44] we use is based on a ViT architecture and has been pre-trained in an adversarial fashion (as opposed to the standard training with an MSE loss) to output more realistic-looking details

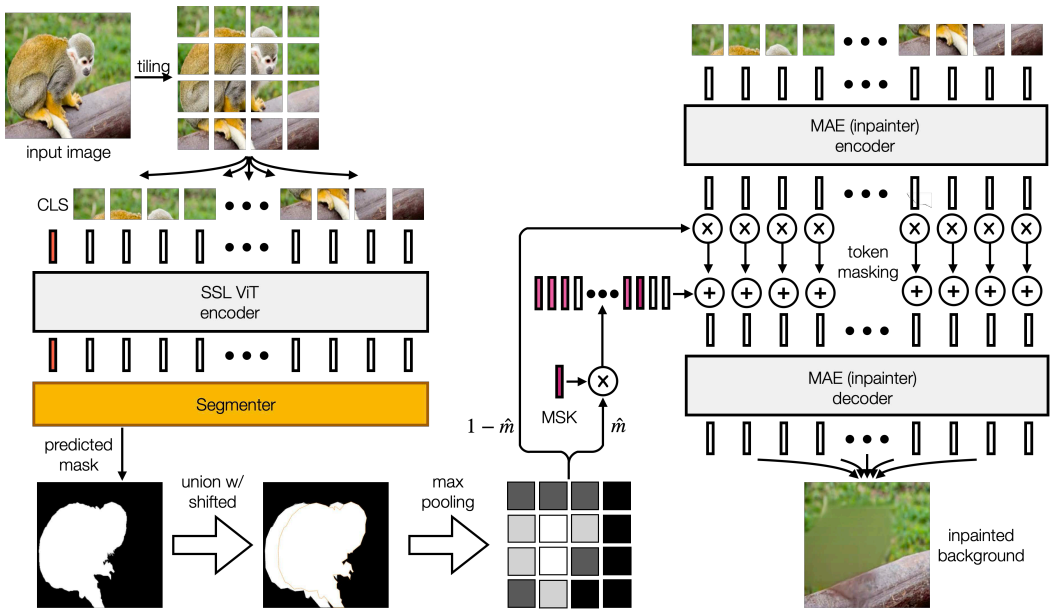


Figure 5.3: (Left) The segmenter is built on top of SSL features from a *frozen* encoder. To define the inpainting region for the background, the predicted mask is shifted and combined with the unshifted mask (bottom left). For better visualization purposes we highlight the edge of the shifted mask, but this does not appear in the actual union of the masks. This mask union is then downsampled to the size of the tile grid via max pooling and denoted $\hat{m}$. (Right) The inpainter is based on a *frozen* MAE. First, it takes all the tiles from the input image and feeds them to the MAE encoder. Second, it takes a convex combination between the encoder embeddings and the MSK learned embedding (but now frozen), where the convex combination coefficients are based on the downsampled mask $\hat{m}$. Finally, this combination is fed to the MAE decoder to generate the inpainted background.

background inpainting artifacts and not because of the artifacts due to an incorrect segmentation (which are exposed by random shifts). To avoid this undesired behavior, we also introduce the real image $\hat{x}_{\delta=0}$. This image has no segmentation artifacts for any mask, but has the same background inpainting artifacts as the fake images (although there is no shift in $\hat{x}_{\delta=0}$, the background inpainting creates artifacts beyond the boundaries of the segmentation mask). Finally, to guide the discriminator to detect repeated patterns (as those caused by incorrect masks, see Figure 5.1) instead of the expected inpainting artifacts, we also add a fake image $\tilde{x}_{\delta}$, where the background has the original foreground.

The segmenter is trained only through the backpropagation from $\hat{x}_\delta$. The details of the segmentation network, the inpainting network, and the adversarial training are explained in the following sections.

5.2.1 Segmenter

Following the recent trend of methods for unsupervised object segmentation [2, 84, 114, 115, 135, 136, 142], we build our method on top of SSL features, in particular, DINO [17] or MAE [44] features. Thus, as a backbone, we adopt the Vision Transformer (ViT) architecture [28]. Following the notation in [115], we split an image $x \in \mathbb{R}^{H\times W\times 3}$ in tiles of size $P \times P$ pixels, for a total of $N = HW/P^2$ tiles (and we assume that H and W are such that H/P and W/P are integers). Each tile is then mapped through a trainable linear layer to an embedding of size d and an additional CLS token is included in the input set (see Figure 5.3 left).

The *segmenter* network is a CNN that takes SSL features as input (e.g., from a pre-trained DINO or MAE encoder), upsamples them and then outputs a mask for the original input image. The final output is generated by applying a sigmoid function to ensure that the mask values are always between 0 and 1. We also ensure a minimum size of the support of the predicted mask by using

$$\mathcal{L}_{\text{min}} = \frac{1}{n}\sum_{i=1}^{n} \max\left\{\theta_{\text{min}} - \sum_p \frac{m^{(i)}[p]}{HW}, 0\right\} \tag{5.1}$$

where n is the number of images in the training dataset, $m^{(i)}$ is the predicted mask from image $x^{(i)}$, p is a pixel location within the image domain, and θ_{min} is a threshold for the minimum mask coverage percentage respectively (in the range $[0, 1]$, where 0 implies that the mask is empty and 1 implies that the mask covers the whole image). Since masks should only take binary values to clearly indicate a segment, we use a loss that encourages $m^{(i)}$ to take either 0 or 1 values

$$\mathcal{L}_{\text{bin}} = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{HW}\sum_p \min\left\{m^{(i)}[p], 1 - m^{(i)}[p]\right\}. \tag{5.2}$$

5.2.2 Differentiable inpainting

Inpainting mask

The main task of MOVE is to predict a segmentation mask that can be used to synthesize a realistic image, where the foreground object is shifted on top of the inpainted background (see Figure 5.1 (e) top and Figure 5.2 top left). Figure 5.3 shows how we use the predicted high-resolution mask for inpainting with MAE. Since MAE performs inpainting by masking or retaining entire patches of $P' \times P'$ pixels, it is

Table 5.1: Inpainting error for a pre-trained MAE on 5000 images from the ImageNet validation set: Feeding a subset of tokens to the encoder (Default) vs soft-masking before the decoder (Modified). Δ is the mean squared error between the inpainted regions for two methods

MAE Model	**Default**	**Modified**	Δ
w/ GAN	0.0683 ± 0.0427	0.0647 ± 0.0398	0.0070 ± 0.0059
w/ MSE	0.0639 ± 0.0411	0.0617 ± 0.0390	0.0055 ± 0.0056

necessary to also split the segmentation mask into a grid of tiles of $P' \times P'$ pixels and to map each tile to a single scalar between 0 and 1. We do that by applying a max pooling operation within each tile and obtain a low-resoluton mask $\hat{m}$, such that $1 - \hat{m}$ does not contain any part of the predicted mask.

However, using max pooling for downsampling might result in inpainting more than necessary due to the artifacts in the mask. To avoid such cases we apply our $\mathcal{L}_{\text{min}}$ and $\mathcal{L}_{\text{bin}}$ losses (eq. (5.1),(5.2)) on the downsampled mask as well. Having a binarization loss on the mask downsampled with max pooling has an extra regularizing effect on the original mask. For example, when all mask pixels in a patch have a value below 0.5, the binarization loss on the max pooling of the mask will push only the largest value towards 0. This creates an asymmetry when the pixels of the mask must be reduced, which prioritizes the largest values. Eventually, however, the application of this loss over multiple iterations will result in pushing all pixels within the patch to 0.

Modified inpainting with MAE

We feed the entire set of image tiles to the MAE encoder and obtain embeddings $\xi_1, \ldots, \xi_N$. Next, for $j = 1, \ldots, N$, we compute the convex combination between the embeddings ξ_j and the learned MSK (masked) token from MAE by using the low res mask $\hat{m}$ as $\hat{\xi}_j = \hat{m}[j] \cdot \xi_{\text{MSK}} + (1 - \hat{m}[j]) \cdot \xi_j$. The MSK token indicates a patch that should be reconstructed. Finally, we feed the new embeddings $\hat{\xi}_j$ in the MAE decoder and reassemble the output tiles back into the inpainted background image $\hat{b}$ (see Figure 5.3 bottom-right). This is in contrast to the typical use of MAE, where only the subset of "visible" tiles is fed as input to the encoder during training (see Figure 2.2, page 28). However, such tile selection operation would make the inpainting not differentiable.

Since in MOVE we feed all the patches to the encoder, it is possible that the encoded embeddings contain information about their neighbors. In particular, there is a risk that the unmasked encoded patches would contain information about the masked patches. If that were the case, the decoder would be able to inpaint the

Figure 5.4: Comparison of MAE sparse input vs differentiable mask inpainting. We show the input and masked input image in the two first columns. For MAE trained with a GAN loss or with an MSE loss we show the reconstructed image when we feed a sparse subset of tokens to the encoder (Def.) and when we feed all the tokens to the encoder and mask only before feeding the embeddings to the decoder (Mod.). No significant difference can be observed between these two reconstruction modalities in terms of missing object reconstruction.

masked object even when the entire object is masked at the decoder input. We show empirically and quantitatively that this is not the case. Using the same pre-trained MAE, we compare the reconstruction error for the original inference vs. our modified soft-masking inference. We run the evaluation on a subset of 5000 images from the ImageNet validation set [25], randomly masking between 80% and 95% of the tokens. We show the mean squared error of the intensity for intensity range $[0; 1]$ in Table 5.1 and comparison of reconstructed images in Figure 5.4 for both MAE trained with a GAN loss or with an MSE loss. We find that the difference in the inpainting error is not significant. Moreover, we observe visually that the reconstructions through the

Modified soft-masking (MOVE) do not show a better reconstruction of the masked patches than in the Default case where the masked patches are not provided to MAE.

5.2.3 Adversarial training

Figure 5.2 shows how we create the images used in the adversarial training. First, we mask the input image with the predicted mask and compose with the inpainted background image, obtaining

$$\hat{x}_\delta[p] = m_\delta[p]\check{x}[p+\delta] + (1 - m_\delta[p])\hat{b}[p], \tag{5.3}$$

where $m_\delta[p] = m[p+\delta]$, $\delta \in [-\Delta H, \Delta H] \times [-\Delta W, \Delta W]$ is a 2D shift, with Δ the maximum shift range (relative to the image size). To make the inpainting artifacts in the no-shift composite image $\hat{x}_{\delta=0}$ more comparable to those in the shifted composite image, we define the background inpainting region as the union between the predicted mask and its shifted version (see Figure 5.3). Thus,

$$\hat{m} = \text{maxpool}_P(1 - (1 - m) \odot (1 - m_\delta)). \tag{5.4}$$

To improve the discriminator's ability to focus on repeated patterns artifacts instead of the expected inpainting artifacts, we additionally create *fake* images with a predicted shifted foreground pasted on top of the autoencoded image, obtaining $\tilde{x}_\delta = \check{x}_\delta \odot m_\delta + \check{x} \odot (1 - m_\delta)$.

The adversarial loss for the discriminator can be written as

$$\mathcal{L}_{\text{advD}} = -\mathbb{E}_{x_R} \min\{0, D(x_R) - 1\} - \mathbb{E}_{x_S} \min\{0, -D(x_S) - 1\} \tag{5.5}$$

where samples for "real" images x_R are the set $\{\check{x}^{(i)}\}_{i=1,\dots,n} \bigcup \{\hat{x}^{(i)}_{\delta=0}\}_{i=1,\dots,n}$ and samples for synthetic images x_S are the set $\{\hat{x}^{(i)}_\delta\}_{i=1,\dots,n} \bigcup \{\tilde{x}^{(i)}_\delta\}_{i=1,\dots,n}$, with uniform random samples $\delta \sim \mathcal{U}_2([-\Delta H, \Delta H] \times [-\Delta W, \Delta W])$ and $\mathbb{E}$ denotes the expectation. To speed up the convergence, we also use the projected discriminator method [109]. For the segmenter, we use instead the standard loss computed on the composite shifted images

$$\mathcal{L}_{\text{advS}} = -\mathbb{E}_{\hat{x}_\delta} D(\hat{x}_\delta). \tag{5.6}$$

Finally, with λ_{min}, λ_{bin} nonnegative hyperparameters, our optimization is the adversarial minimization

$$S^* = \arg\min_S \mathcal{L}_{\text{advS}} + \lambda_{\text{min}}\mathcal{L}_{\text{min}} + \lambda_{\text{bin}}\mathcal{L}_{\text{bin}} \tag{5.7}$$

$$\text{subject to } D^* = \arg\min_D \mathcal{L}_{\text{advD}}. \tag{5.8}$$

5.3 Implementation

Except for the ablation studies, in all our experiments we use a self-supervised DINO [17] ViT-S/8 transformer pre-trained on ImageNet [25] as an SSL feature extractor. We take the output of the penultimate transformer block of DINO as the feature tokens with $P = 8$ and feed them to the segmenter. Our segmenter is a small upsampling convolutional neural network. It assembles the DINO features into a grid of size $H/P \times W/P$ and processes them with 3 upsampling blocks, so that the output matches the input image resolution. Each upsampling block first performs a 2×2 nearest upsampling, followed by a 3×3 convolutional layer with padding, batch normalization [47] and a LeakyReLU activation function. We add an additional block without upsampling followed by a linear projection to 1 channel, representing the mask. Our inpainting network is a ViT-L/16 transformer pre-trained on ImageNet as a self-supervised Masked Autoencoder (MAE) [44] with an adversarial loss to increase the details of the reconstructed images. For the discriminator we use the Projected Discriminator [109] in its standard setting, but we only use *color* differentiable augmentation. For the training we use random resized crops of size 224 with a scale in range $(0.9, 1)$ and aspect ratio $(3/4, 4/3)$. We set the minimum mask area $\theta_{\text{min}} = 0.05$, the minimum loss coefficient $\lambda_{\text{min}} = 100$ and we linearly ramp up the binarization loss coefficient λ_{bin} from 0 to 12.5 over the first 2500 segmenter iterations. We use the shift range $\Delta = 1/8$. We train the segmenter by alternatively minimizing the discriminator loss and the segmenter losses. Both are trained with a learning rate of 0.0002 and an Adam [61] optimizer with betas $= (0, 0.99)$ for the discriminator and $(0.9, 0.95)$ for the segmenter. We implemented our experiments in PyTorch [101]. We train our model for 80 epochs with a batch size of 32 on a single NVIDIA GeForce 3090Ti GPU with 24GB of memory.

5.4 Experiments

5.4.1 Unsupervised saliency segmentation

Datasets

We train our main model using the train split of the DUTS dataset (DUTS-TR) [131], containing 10,553 images of scenes and objects of varying sizes and appearances. We emphasize that we only use the images without the corresponding ground truth. For comparison, we evaluate our approach on three saliency detection datasets: the test set of DUTS (5,019 images), DUT-OMRON [139] (5,168 images) and ECSSD [113] (1,000 images). We report three standard metrics: pixel mask accuracy (Acc), Intersection over Union between the predicted and ground truth mask IoU $= \frac{|\mathrm{m}_{\text{pred}} \cap \mathrm{m}_{\text{gt}}|}{|\mathrm{m}_{\text{pred}} \cup \mathrm{m}_{\text{gt}}|}$, $\max F_\beta$,

Table 5.2: Comparison to the unsupervised saliency detection methods on 3 benchmarks

Model	DUT-OMRON [139]			DUTS-TE [131]			ECSSD [113]		
	Acc	IoU	maxF_β	Acc	IoU	maxF_β	Acc	IoU	maxF_β
HS [138]	.843	.433	.561	.826	.369	.504	.847	.508	.673
wCtr [156]	.838	.416	.541	.835	.392	.522	.862	.517	.684
WSC [73]	.865	.387	.523	.862	.384	.528	.852	.498	.683
DeepUSPS [92]	.779	.305	.414	.773	.305	.425	.795	.440	.584
SelfMask pseudo* [114]	.811	.403	-	.845	.466	-	.893	.646	-
BigBiGAN [129]	.856	.453	.549	.878	.498	.608	.899	.672	.782
E-BigBiGAN [129]	.860	.464	.563	.882	.511	.624	.906	.684	.797
Melas-Kyriazi et al. [85]	.883	.509	-	.893	.528	-	.915	.713	-
LOST [115]	.797	.410	.473	.871	.518	.611	.895	.654	.758
Deep Spectral [84]	-	.567	-	-	.514	-	-	.733	-
TokenCut [136]	.880	.533	.600	.903	.576	.672	.918	.712	.803
FreeSOLO [135]	.909	.560	.684	.924	.613	.750	.917	.703	.858
MOVE (Ours)	**.923**	**.615**	**.712**	**.950**	**.713**	**.815**	**.954**	**.830**	**.916**
LOST [115] + Bilateral	.818	.489	.578	.887	.572	.697	.916	.723	.837
TokenCut [136] + Bilateral	.897	.618	.697	.914	.624	.755	.934	.772	.874
MOVE (Ours) + Bilateral	**.931**	**.636**	**.734**	**.951**	**.687**	**.821**	**.953**	**.801**	**.916**
SelfMask on pseudo* [114]	.923	.609	.733	.938	.648	.789	.943	.779	.894
SelfMask on pseudo* [114] + Bilateral	**.939**	**.677**	**.774**	.949	.694	.819	.951	.803	.911
SelfMask on MOVE (Ours)	.933	.666	.756	**.954**	**.728**	**.829**	**.956**	**.835**	**.921**
SelfMask on MOVE (Ours) + Bilateral	.937	.665	.766	.952	.687	.827	.952	.800	.917

*We found that SelfMask's max F_β metric was computed with an optimal threshold for each image instead of the entire dataset as in other methods; we re-evaluated their model for a fair comparison

where $F_\beta = \frac{(1+\beta^2)\text{Precision}\times\text{Recall}}{\beta^2\text{Precision}+\text{Recall}}$ for $\beta^2 = 0.3$; the max F_β is the score for the single optimal threshold on a whole dataset. Additionally, we report the IoU on the test split [18] of CUB-200-2011 (CUB-Birds) [130] dataset.

Evaluation

We train our segmenter in an adversarial manner as specified in sections 5.2 and 5.3 and evaluate it on the test datasets. We compare with other methods in Table 5.2. Note that without any type of post-processing of our predicted masks, we surpass all other methods by a significant margin. We also follow [114, 136] and further refine our masks with a bilateral solver [7]. Since the bilateral solver only marginally improves

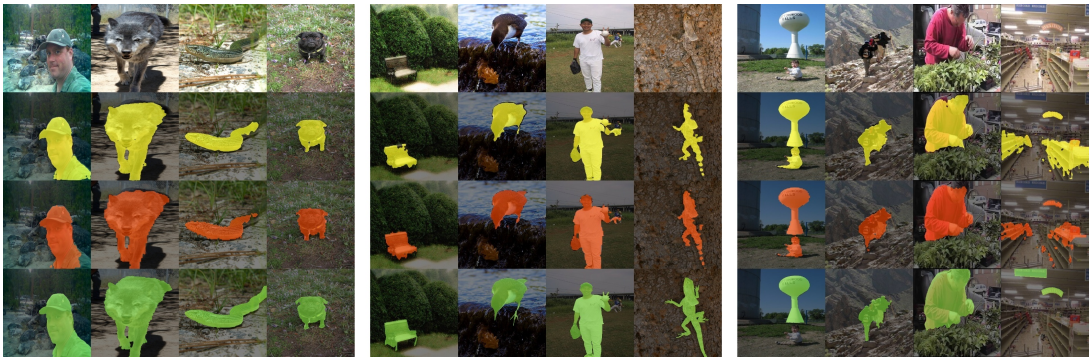


Figure 5.5: Qualitative evaluation of MOVE on ECSSD, DUTS-TE and DUT-OMRON. First row: input image; second row: MOVE; third row: SelfMask on MOVE; last row: ground truth. Best viewed in color. More examples in Figures 5.8, 5.9, 5.10.

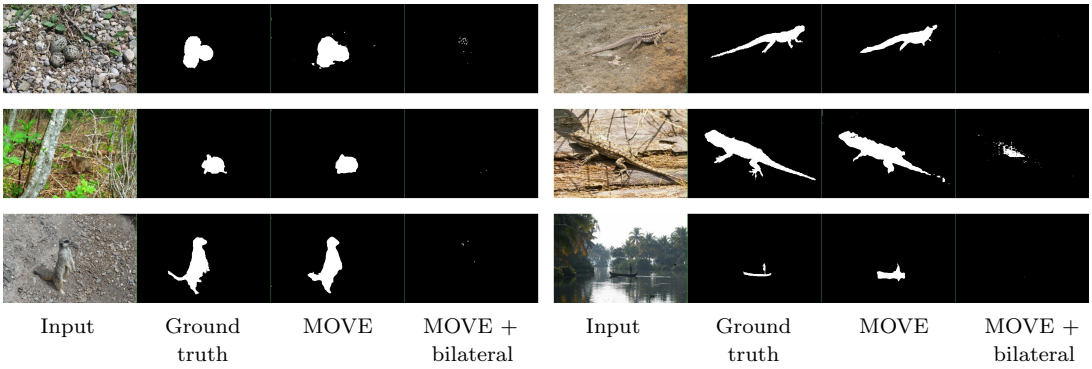


Figure 5.6: A refinement with the bilateral solver might cause the shrinking of valid predicted masks.

or even decreases the quality of our segmentation, we conclude that our predicted masks are already very accurate. Using the bilateral solver might also inadvertently discard correct, but fragmented segmentations, as we show in Figure 5.6. Next, we extract the predicted unsupervised masks from the DUTS-TR dataset and use them as pseudo ground-truth to train a class-agnostic segmenter. We use the same architecture (a MaskFormer [21]) and training scheme as SelfMask [114]. We then evaluate again on the saliency prediction datasets. Without additional pre-processing our method surpasses or is on par with the SotA across all metrics and datasets. While additional processing with the bilateral solver seems to benefit SelfMask [114], it mostly hurts the performance of our method. Figure 5.5 shows qualitative results of our method. Finally, we evaluate our method on the test set of CUB-Birds dataset. Additionally, we train our model on the train split of CUB-Birds dataset and run the same evaluation. We present the comparison with other methods in Table 5.3 and show that we achieve

Table 5.3: Comparison of unsupervised segmentation methods on the CUB-200-2011 test set. MOVE* was trained on the CUB-200-2011 train set, while MOVE was trained on DUTS-TR

Method	**IoU**
PerturbGAN [10]	0.360
ReDO [18]	0.426
IEM [110]	0.551
Melas-Kyriazi [85]	0.664
Voynov [129]	0.683
Voynov-E [129]	0.710
Deep Spectral [84]	0.769
MOVE*	**0.814**
MOVE	**0.858**

state-of-the-art performance. In Figures 5.8, 5.9, 5.10 we show more segmentation results of MOVE on DUTS-TE, DUT-OMRON and ECSSD.

5.4.2 Single-object discovery

Datasets

We evaluate our trained model (see section 5.4.1) on 3 typical single-object discovery benchmarks: the train split of COCO20K [76, 127] and the trainval splits of VOC07 [31] and VOC12 [32]. Following [22, 26, 115, 117, 126–128, 136], we report the *Correct Localization* metric (CorLoc), *i.e.*, the percentage of images in which the predicted single bounding box matches at least one of the ground truth boxes with IoU > 0.5.

Evaluation

Since our method tends to produce a single segmentation mask for multiple objects in the scene, we separate the objects by detecting connected components via OpenCV [13]. We then convert the separate masks to bounding boxes and choose the biggest one as our prediction for the given image. In Table 5.4, we compare MOVE with other unsupervised methods and we show that just by using processed masks from our method we achieve state-of-the-art results on all three datasets, outperforming even methods that used their bounding boxes to train a Class Agnostic Detector (CAD). We present qualitative results for object detection in Figure 5.7. We also follow the practice of [115, 136] and use our predicted bounding boxes as pseudo-ground truth for training the CAD on each of the evaluation datasets. To train the detector, we use either

Table 5.4: Comparisons for unsupervised single object discovery. We compare MOVE to SotA object discovery methods on VOC07 [31], VOC12 [32] and COCO20K [76, 127] datasets. Models are evaluated with the CorLoc metric. +CAD indicates training a second stage class-agnostic detector with unsupervised "pseudo-boxes" labels. (↑ **z**) indicates an improvement of z over prior SotA

Method	**VOC07**	**VOC12**	**COCO20K**
Selective Search [115, 122]	18.8	20.9	16.0
EdgeBoxes [115, 157]	31.1	31.6	28.8
Kim et al. [60, 115]	43.9	46.4	35.1
Zhange et al. [115, 152]	46.2	50.5	34.8
DDT+ [115, 137]	50.2	53.1	38.2
rOSD [115, 127]	54.5	55.3	48.5
LOD [115, 128]	53.6	55.1	48.5
DINO-seg [17, 115]	45.8	46.2	42.1
FreeSOLO [135]	56.1	56.7	52.8
LOST [115]	61.9	64.0	50.7
Deep Spectral [84]	62.7	66.4	52.2
TokenCut [136]	68.8	72.1	58.8
MOVE (Ours)	**76.0** (↑ **7.2**)	**78.8** (↑ **6.7**)	**66.6** (↑ **7.8**)
LOD + CAD[115]	56.3	61.6	52.7
rOSD + CAD [115]	58.3	62.3	53.0
LOST + CAD [115]	65.7	70.4	57.5
TokenCut + CAD [136]	71.4	75.3	62.6
MOVE (Ours) + CAD	77.1	80.3	69.1
MOVE (Ours) Multi + CAD	**77.5** (↑ **6.1**)	**81.5** (↑ **6.2**)	**71.9** (↑ **9.3**)

the largest or all the bounding boxes (*Multi*) that we obtained from the connected components analysis and after filtering those that have an area smaller than 1% of the image. For the evaluation we take the bounding box with the highest prediction confidence, as done in [115, 136]. We use the exact same architecture and training scheme as our competitors for a fair comparison. Training with a single bounding box improves the performance of our method, while training with multiple ones gives it a significant additional boost. In Figures 5.11, 5.12, 5.13 we show more object detection results of MOVE on VOC07, VOC12, and COCO20k.

Unsupervised class-agnostic object detection

We evaluate our unsupervised object detection model trained on COCO20K with CAD post-training and compare it with SotA on unsupervised class-agnostic object detection.

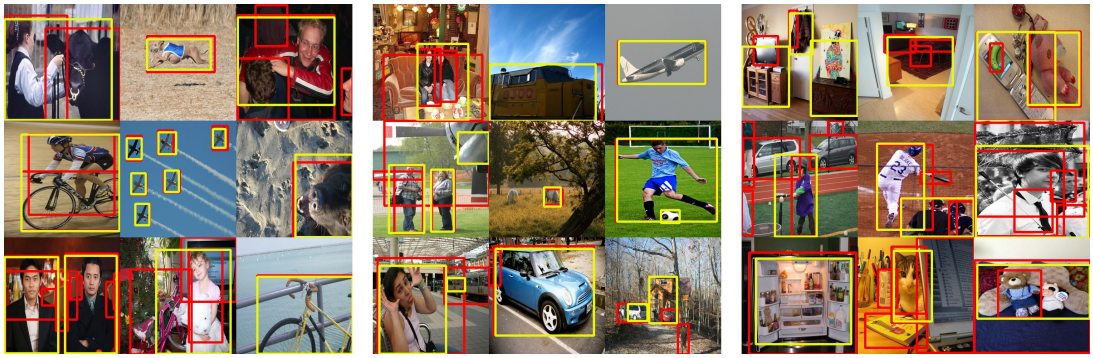


Figure 5.7: Qualitative evaluation of object detection of MOVE on VOC07, VOC12 and COCO20k. Red is the ground truth, yellow is our prediction. More examples in Figures 5.11, 5.12, 5.13.

Table 5.5: Unsupervised class-agnostic object detection on MS COCO `val2017`. Compared results are taken directly from FreeSOLO [135]

Method	AP_{50}	AP_{75}	AP	AR_1	AR_{10}	AR_{100}
Sel. Search [122]	0.5	0.1	0.2	0.2	1.5	10.9
DETReg [5]	3.1	0.6	1.0	0.6	3.6	12.7
FreeSOLO [135]	12.2	4.2	5.5	4.6	11.4	15.3
MOVE (Ours)	**19.0**	**6.5**	**8.2**	**5.7**	**13.6**	**15.9**

In Table 5.5, we evaluate MOVE on COCOval2017 and report Average Precision (AP) and Average Recall (AR), as in [135]. MOVE yields a remarkable relative improvement over the AP SotA of 50% on average.

5.4.3 Ablation study

We perform ablation experiments on the validation split (500 images) of HKU-IS [72] to validate the relative importance of the components of our segmentation approach. For the ablation study, we train each model for 80 epochs on DUTS-TR. We report the IoU in Table 5.6. Our baseline model trained with 3 different seeds gives a mean IoU 0.818 with std = 0.008. Thus we only report results for a single run in all experiments.
Mask losses. We validate the importance of the mask losses: minimum mask area, binarization, and losses on downsampled max-pooled and avg-pooled masks. We find that the minimum area loss is necessary for our method to work, otherwise there is no incentive to produce anything other than empty masks. Removing the binarization loss

Table 5.6: Ablation study. Models evaluated on HKU-IS-val

Setting	**IoU**
Baseline (shift $^{2}/_{16}$)	**0.819**
no min. mask	0.000
no binarization loss	0.774
no pooled mask losses	0.811
no shift	0.000
shift $^{1}/_{16}$	0.751
shift $^{3}/_{16}$	0.799
shift $^{4}/_{16}$	0.704
disc. fake inputs: composed	0.789
disc. real inputs: x + comp. w/o shift	0.740
disc. real inputs: comp. w/o shift	0.031
disc. real inputs: x_{ae}	0.000
non-diff inpainter	0.314
MSE MAE	0.817
MAE feature extractor	0.783
ImageNet100 dataset	0.815

or mask losses on the downsampled masks makes the masks noisier, which negatively affects the results.

Shift range. We evaluate different ranges of the random shift δ. A small range $\Delta = {}^{1}/_{16}$ makes it more challenging for the discriminator to detect inconsistencies at the border of objects. Larger shifts may cause objects to go out of the image boundaries ($\Delta = {}^{3}/_{16}, {}^{4}/_{16}$) and thus reduce the feedback at the object boundary to the segmenter. For $\Delta = 0$ (no-shift) the only possible discriminator inputs are composed images without a shift as fake and autoencoded images as real. There is no incentive to produce any meaningful masks in this case.

Discriminator inputs. In our baseline model, we feed both composed images with no-shift and real images autoencoded with MAE as *real samples*, and composed images with a shift and autoencoded images with copy-pasting of a predicted masked object as *fake samples* for the discriminator training. We test the case DISC. REAL x + COMP. W/O SHIFT , where we feed to the discriminator real images without autoencoding. In this case, the discriminator can detect the artifacts of MAE instead of focusing on inconsistencies resulting from an incorrect mask. In DISC. REAL x_{ae} we only feed the autoencoded images as real. Here, the discriminator can focus on the mismatch from the inpainting artifacts and encourages the segmenter to output empty masks, where

no inpainting is done. If we only feed the composite non-shifted images (DISC. REAL COMP W/O SHIFT), the artifacts resulting from an incorrect masks cannot be fixed, because there is no reference of what the real images look like. In DISC. FAKE INPUTS: COMPOSED we only feed the composed image as fake to the discriminator and omit the real image with a copy-pasted predicted masked object, which slightly degrades the performance.

Non-differentiable inpainter. We evaluate the use of hard thresholded downsampled masks as input to the background inpainter. In this case the only feedback for the masks comes from the composition of the images. We find it to be insufficient for the segmenter to learn any meaningful masks.

Inpainter model. We substitute the MAE trained with a GAN loss with a MAE that was trained only to reconstruct missing patches with a Mean Squared Error (MSE) loss. Since this model was trained to only reconstruct the missing patches and not the entire image, we construct the inpainted background by composing the inpainted part with the real image: $\hat{m}_{up} = \text{upsample}_{16}(\hat{m})$; $\hat{b} := x \odot (1 - \hat{m}_{up}) + \hat{b} \odot \hat{m}_{up}$. Consequently, we do not use autoencoding when creating the discriminator inputs. We find this model to perform competitively.

Feature extractor. We train the model using the features provided by MAE encoder instead of a separate DINO model. In this case we adapted the segmenter architecture and added one more upsampling block, since MAE takes patches of size $P = 16$ (instead DINO has $P = 8$). We find that with these features we are able to train a competitive segmenter.

ImageNet100 dataset. We train our model on the ImageNet100 dataset [121], with 131,689 images from 100 randomly selected ImageNet [25] classes. Since this dataset is much bigger than DUTS-TR, we adapt our segmenter by adding an additional convolutional layer in each upsampling block (see section 5.3) and train the model for 8 epochs. The results are comparable to the DUTS-TR dataset.

5.5 Discussion

We have introduced MOVE, a novel self-supervised method for object segmentation that exploits the synthesis of images where objects are randomly shifted. MOVE improves the state of the art in object saliency segmentation, unsupervised single object discovery, and unsupervised class agnostic object detection by significant margins. Our ablations show that movability is a strong supervision signal that can be robustly exploited as a pseudo-task for self-supervised object segmentation. We believe that our approach can be further scaled by exploring different architectures and larger datasets.

Despite impressive results, our method has certain limitations. Movability alone may not suffice to unambiguously identify an object. Indeed, the method can segment

any combination of multiple objects. To address this we use a post-processing algorithm to find connected components, but there is no guarantee that all objects have been segmented. Another challenge arises when shifts do not expose artifacts against uniform backgrounds, for example, when viewing the sky or in underwater scenes.

Figure 5.8: Sample segmentation results on ECSSD.

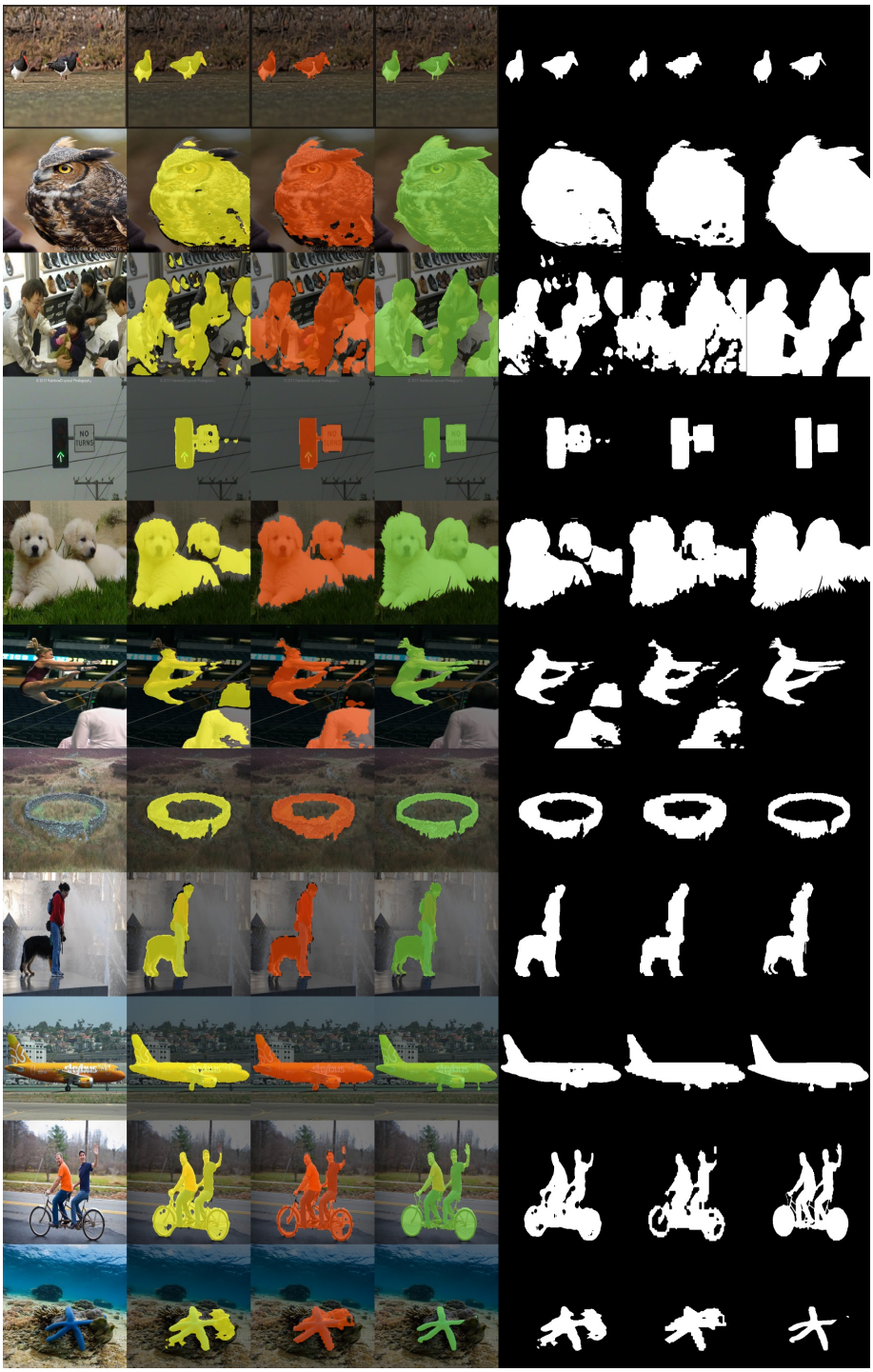


Figure 5.9: Sample segmentation results on DUTS-TE.

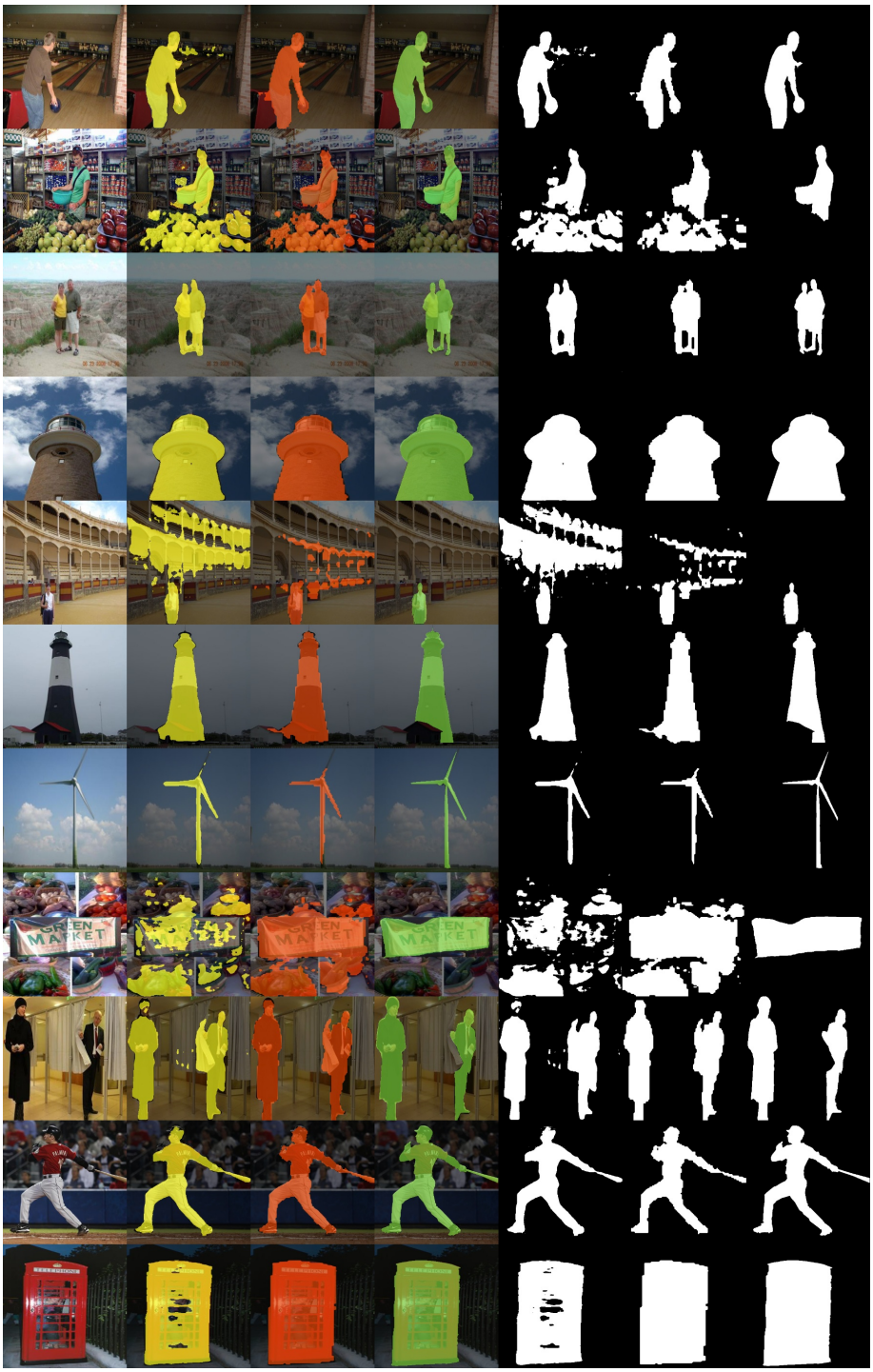


Figure 5.10: Sample segmentation results on DUT-OMRON.

Figure 5.11: Sample detection results on VOC07.

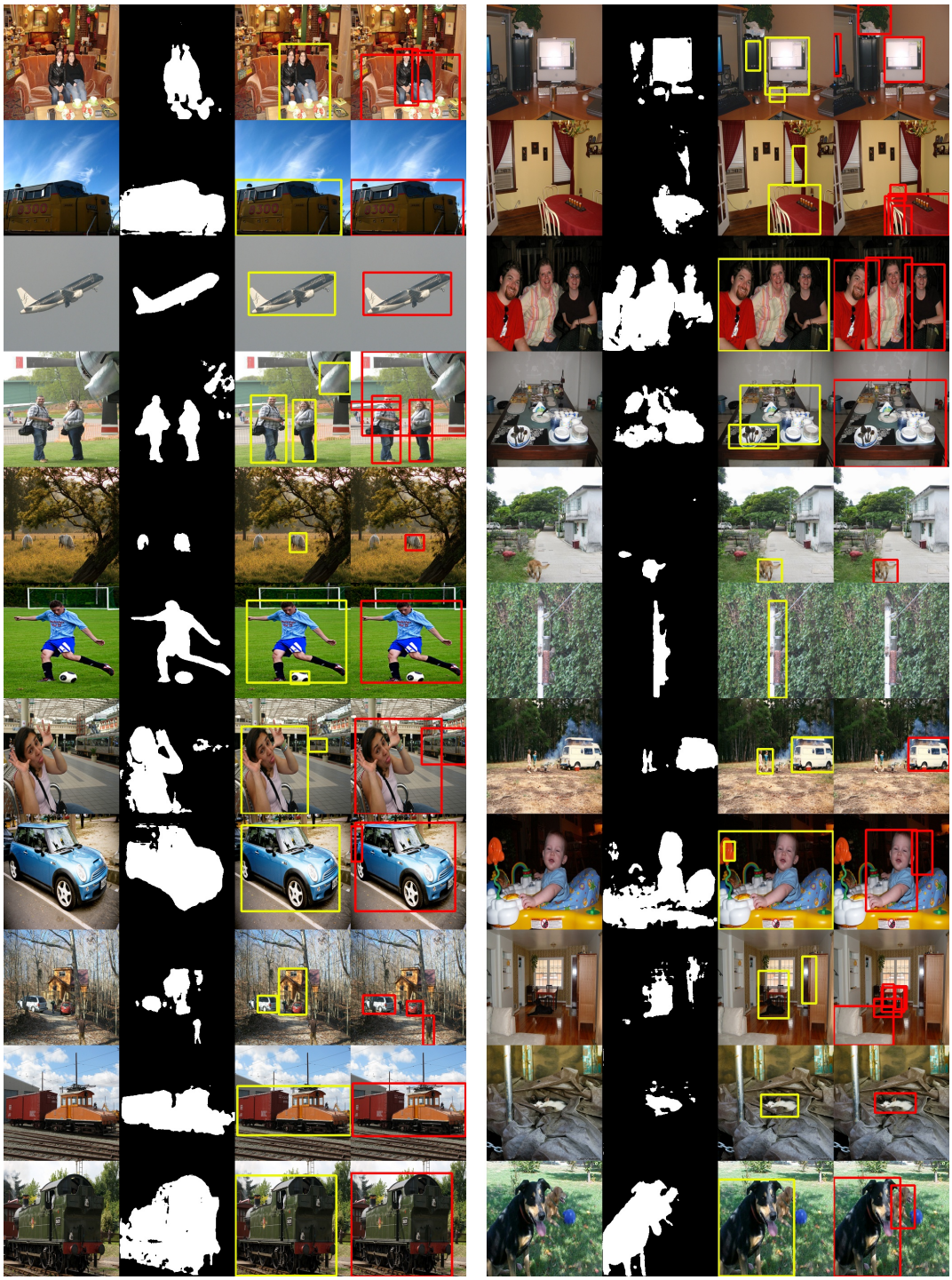


Figure 5.12: Sample detection results on VOC12.

Figure 5.13: Sample detection results on COCO20k.

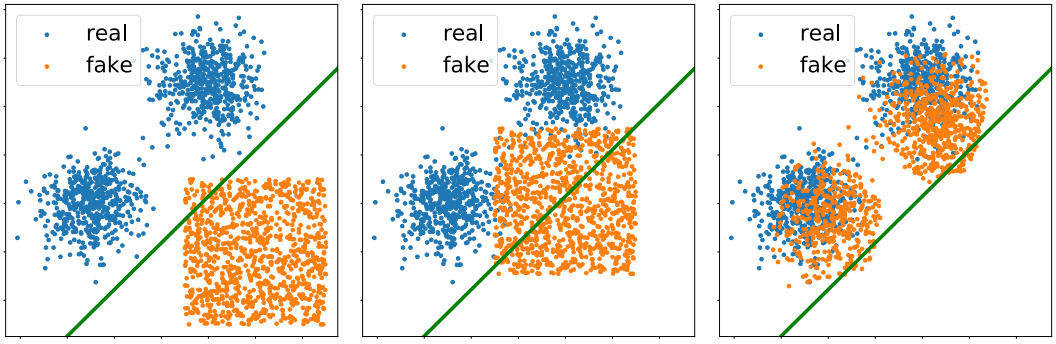
Chapter 6

Generative Adversarial Learning via Kernel Density Discrimination

Generative Adversarial Networks, or GANs, have been widely successful thanks to several breakthroughs in the design of the generator and discriminator architectures [14, 55, 148], of the loss functions [4, 53, 145] and regularization methods [53, 82, 90, 149] (see Section 2.4, page 29). Yet, the training of generative models is not straightforward and can be still prone to mode collapse [79, 120, 144] or the inability to capture long-range statistics in the data, which leads to visible artifacts [75, 148].

We introduce the Kernel Density Discrimination GAN (KDD GAN), a novel method for generative adversarial learning. KDD GAN formulates the training as a likelihood ratio optimization problem where the data distributions are written explicitly via (local) Kernel Density Estimates (KDE). This is inspired by the recent progress in contrastive learning and its relation to KDE.

One key assumption in the basic formulation of adversarial learning of [35] is that the generator network should compete with an optimal discriminator, that is, a classifier that can tell real from generated data apart if any of their statistics does not match. Thus, the general wisdom is that the more powerful the discriminator is, the better the generator trains. Given that training models with contrastive losses yields better performance than training with cross-entropy losses [59], and that contrastive learning can be seen as introducing Kernel Density Estimate (KDE) approximations of the data distribution [132], we propose to train the discriminative and generative models through a KDE approximation of the likelihood ratio loss. Moreover, this approach ensures that the loss defines a valid statistical divergence between the distributions of the real and generated data at all times. In contrast, the loss used to train state of the art



(a) Initial state (b) Hinge loss (c) KDD loss

Figure 6.1: **Illustration of the difference between the hinge loss and KDD loss during the generator update.** The blue and orange point clouds represent the discriminator features of the real and fake samples. The initial positions of the samples are shown in Fig. 6.1a. The green line in all three sub-figures represents the decision boundary associated with the optimal linear classifier separating the two distributions at the initial state. Fig. 6.1b and Fig. 6.1c show the updated positions of the fake samples using the Hinge loss and KDD loss respectively. The generator update via the KDD loss leads to a more detailed overlap.

generative adversarial networks corresponds to a known statistical divergence between distributions of real and fake data only when at the saddle point of the min-max game. Our analysis shows that the gradients of the proposed loss are better behaved than those of the hinge loss (as defined, for example, by [89]). We propose a KDE defined directly in feature space, so that non-invertible features are allowed. Our method includes a much broader set of discriminator solutions than in the binary classification task of the original GAN formulation. In fact, in the KDE approach the features are no longer optimized for linear separability, but for the more general discrimination of distributions in the feature space. This can be seen clearly in Fig. 6.1 for 2D point clouds. We call our method *Kernel Density Discrimination GAN* (KDD GAN).

Constributions. We propose a novel KDD loss and provide a theoretical proof that KDD GAN converges to the unique equilibrium point, where the distribution of generated samples matches that of real data. KDD GAN outperforms BigGAN [14] (which we use as a backbone) on CIFAR10 [63] and Tiny ImageNet [66] by more than 10% in the FID and IS metrics. The proposed KDD loss is flexible and when combined with other methods as a regularizer improves the training in terms of FID and IS on CIFAR10, Tiny ImageNet, and ImageNet 64×64, which has images scaled to 64×64 pixels (derived from [25]). The implementation of KDD GAN is on par

with conventional hinge loss training [89] in terms of the computational load and the memory footprint.

6.1 Background

In attempts to address the limitations of GANs (see Section 2.4, page 29), other kernel-based GANs have been proposed previously. [116] explore the idea of using a non-parametric estimate of the Jensen-Shanon Divergence and use KDEs for the purpose of training GANs. This idea is very similar to the one explored in this work. The main differences are that Kernel GAN [116] computes its KDEs in the image space and for a simpler selection of datasets; also it requires an additional auto-encoding constraint and computing the KDEs in feature space for more complex datasets. Alternatively, MMD-GAN [71] and its variants such as [133] explore the idea of matching the two distributions at hand by optimizing the Maximum Mean Discrepancy defined by the chosen kernel. Although the improved MMD introduced in [133] bears a few similarities to our work in terms of having both attractive and repulsive loss terms, the two frameworks are fundamentally different. Our KDD-GAN aims at matching the two distributions in the feature space defined by the discriminator, while MMD-GAN and its variants aim at minimizing the maximum mean discrepancy in the RKHS defined by the kernel choice.

6.2 Kernel Density Discrimination

Let $S_r = \{x_r^{(1)}, \dots, x_r^{(m)}\}$ be a dataset of m image samples $x_r^{(i)} \in \mathbb{R}^d$, which we call *real data*. They are the instances of a probability density function (pdf) p_r, which we call the *real data* pdf. We aim to build a generative model that maps zero-mean Gaussian samples to images, and such that they also follow the real data distribution. To distinguish real from generated samples, we denote the dataset of generated data by S_g, a generated image sample by x_g, and the *generated data* pdf by p_g.

We build our generative model through adversarial learning as in the pioneering work of [35], and thus work with a *discriminator* network D and a *generator* network G. Then, generative adversarial learning can be cast as the following bilevel optimization problem

$$\min_G \ \mathcal{L}_G(D^*, G) \quad \text{s.t.} \quad D^* = \arg\min_D \mathcal{L}_D(D, G), \tag{6.1}$$

where the optimization in G and D is implemented as the optimization with respect to the parameters of the neural networks implementing G and D. In the case of hinge

loss optimization (see *e.g.*, [89]), the losses in eq. (6.1) are defined as

$$\mathcal{L}_D^{\text{Hinge}}(D,G) = \frac{1}{|S_g|}\sum_{x_g\in S_g}\max\{0, 1+D(x_g)\} + \frac{1}{|S_r|}\sum_{x_r\in S_r}\max\{0,1-D(x_r)\} \tag{6.2}$$

$$\mathcal{L}_G^{\text{Hinge}}(D^*,G) = \frac{1}{|S_g|}\sum_{x_g\in S_g} -D^*(x_g), \tag{6.3}$$

which rely on the assumption that the discriminator takes the form of

$$D^* = \log p_r(x) - \log p_g(x). \tag{6.4}$$

In our approach, we would like instead to explicitly approximate the form $\log \frac{p_r(x)}{p_g(x)}$. The main advantage of having this form is that it is a well-defined divergence between distributions. Thus, it defines a valid gradient for the generator at all times, up to the errors due to the chosen approximation.

We propose to approximate $p_r(x)$ and $p_g(x)$ in the definition of $D(x)$ with Kernel Density Estimates (KDE). The kernels are defined in feature space and the feature mappings are estimated during training. A simple way to ensure that at the convergence of the bi-level optimization (*i.e.*, when the minima have been reached) the real and fake pdfs match, is to require the invertibility of the feature mappings. Invertibility is the same requirement of Normalizing Flows (see, *e.g.*, [62]) and thus one would have to follow similar restrictions in the neural architectures used to compute the features. However, training invertible neural networks is not easy and, as we argue here below, also not necessary. To simplify the training of the generative model, we propose instead to use KDEs in feature space $\phi : \mathbb{R}^d \to \mathbb{R}^K$ defined by the last layer of the discriminator, and to allow the feature mapping to be non-invertible. Thus, we aim to match the push-forward measures $\phi_* p_r$ and $\phi_* p_g$, which we denote by $\hat{p}_r^{\phi}$ and $\hat{p}_g^{\phi}$ respectively.

We write the losses for KDD GAN explicitly as

$$\mathcal{L}_D^{KDD}(\phi,G) = \frac{1}{|S_g|}\sum_{x_g\in S_g}\max\left\{0, 1+\log\frac{\hat{p}_r^{\phi}(x_g)}{\hat{p}_g^{\phi}(x_g)}\right\} + \frac{1}{|S_r|}\sum_{x_r\in S_r}\max\left\{0,1-\log\frac{\hat{p}_r^{\phi}(x_r)}{\hat{p}_g^{\phi}(x_r)}\right\} \tag{6.5}$$

$$\mathcal{L}_G^{KDD}(\phi^*,G) = \frac{1}{|S_g|}\sum_{x_g\in S_g} -\log\frac{\hat{p}_r^{\phi^*}(x_g)}{\hat{p}_g^{\phi^*}(x_g)}, \tag{6.6}$$

by approximating the push-forward measures of the pdfs p_r and p_g via the following KDEs in feature space

$$\hat{p}_r^{\phi}(\xi) = \frac{1}{|S_r|} \sum_{x_r \in S_r} \mathcal{K}_\tau(\phi(x_r), \xi), \tag{6.7}$$

$$\hat{p}_g^{\phi}(\xi) = \frac{1}{|S_g|} \sum_{x_g \in S_g} \mathcal{K}_\tau(\phi(x_g), \xi) \tag{6.8}$$

where

$$\mathcal{K}_\tau(\phi(x), \xi) = \frac{1}{Z} e^{\frac{\langle \phi(x), \xi \rangle}{\tau}} \tag{6.9}$$

is a positive kernel that integrates to 1 in ξ, $\tau > 0$ is a temperature parameter that relates to the spread of each kernel, $|S|$ is the cardinality of S, and Z is the normalization constant (this becomes irrelevant as it cancels out in the ratios in $\mathcal{L}_D(\phi, G)$ and $\mathcal{L}_G(\phi^*, G)$). The features $\phi(x)$ are L^2-normalized through the projection on the unit hypersphere, *i.e.*, $|\phi(x)|_2 = 1$. Essentially, we assume that the features are samples of a mixture of von Mises-Fisher (vMF) distributions, where all concentration parameters are equal to $1/\tau$.

As mentioned above, the convergence of KDD GAN does not need the invertibility of the feature mapping ϕ. We show this result formally in Theorem 1 and address the invertibility in Lemma 1.

Lemma 1. *Let p_r and p_g be two distributions over $\mathbb{R}^d$. Given a positive integer K we have that $p_r = p_g \Leftrightarrow \forall\, \phi : \mathbb{R}^d \to \mathbb{R}^K, \hat{p}_r^{\phi} = \hat{p}_g^{\phi}$.*

Proof of Lemma 1. $p_r = p_g \Rightarrow \forall \phi : \mathbb{R}^d \to \mathbb{R}^K, \hat{p}_r^{\phi} = \hat{p}_g^{\phi}$ is trivial since $\{\phi(x), x \sim p_r\}$ and $\{\phi(x), x \sim p_g\}$ are the same set when $p_r = p_g$.

Assume $p_r \neq p_g$. Then, there exists an optimal binary classifier c, whose accuracy is above chance level, *i.e.*, $P(\{c(x) = 1, x \sim p_r\}) > \frac{1}{2}$. We can define a mapping $\phi(x) := c(x)\mathbb{1}_K$ where $\mathbb{1}_K$ is the vector of ones in $\mathbb{R}^K$. In this case, we obtain $E_{x \sim p_r}[\phi(x)^T \mathbb{1}_K] = E_{x \sim p_r}[c(x)]K > \frac{K}{2}$ and $E_{x \sim p_g}[\phi(x)^T \mathbb{1}_K] = E_{x \sim p_g}[c(x)]K < \frac{K}{2}$. This implies that first moments of $\hat{p}_r^{\phi}$ and $\hat{p}_g^{\phi}$ are different, thus $\hat{p}_r^{\phi} \neq \hat{p}_g^{\phi}$. Therefore, by contradiction, $\forall\, \phi : \mathbb{R}^d \to \mathbb{R}^K,\ \hat{p}_r^{\phi} = \hat{p}_g^{\phi} \Rightarrow p_r = p_g$ □.

Theorem 1. *$p_g = p_r$ is the unique equilibrium point for KDD GAN.*

Proof of Theorem 1. Let us assume there exists an equilibrium point (ϕ, G) such that $p_r \neq p_g$. Then, we have two cases: either $\hat{p}_r^{\phi} = \hat{p}_g^{\phi}$ or $\hat{p}_r^{\phi} \neq \hat{p}_g^{\phi}$. Let us assume $\hat{p}_r^{\phi} = \hat{p}_g^{\phi}$. Then, according to Lemma 1, there exists a φ such that $\hat{p}_r^{\varphi} \neq \hat{p}_g^{\varphi}$; *i.e.*, ϕ is not an equilibrium point of $\mathcal{L}_D^{KDD}$ Now, let us assume instead that $\hat{p}_r^{\phi} \neq \hat{p}_g^{\phi}$, then G is not an equilibrium point of $\mathcal{L}_G^{KDD}$ □.

6.2.1 Improving KDE through Data Augmentation

The KDEs in eq. (6.8) are mixtures of von Mises-Fisher distributions centered around a set of *anchor points*. In the KDE approximation we cannot use the entire dataset S_r as anchor points, because it would be too computationally demanding. Instead, at each iteration of the training procedure we sample a subset (a minibatch) and use this as anchor points. A fundamental requirement of the KDE approximation is that these sets should be representative of the true distributions p_r or p_g. However, KDE approximations are in general very poor with high dimensional data, as they require a very large number of anchor points. This is because only the kernels that correspond to anchor points that are "similar" to the input sample dominate in the KDE. However, the likelihood of finding these anchor points through uniform sampling becomes extremely small as we grow in data dimensionality.

One way around this problem is to enrich the set of anchors using data augmentations. Provided that the chosen data augmentation does not produce samples outside the manifold of natural images, this allows us to obtain anchor points that are close enough to give a meaningful KDE.

For similar reasons, we use a *leave one out* KDE, where we remove the anchor point from the set S_r or S_g that the KDE is being evaluated on. This avoids a bias towards the unlikely case where we sample exactly a point in the anchor point set. We show experimentally that these KDE implementation details are indeed quite important in boosting the effectiveness of the proposed approach.

6.2.2 Loss Analysis

We analyze the impact of the proposed loss on the generator training and compare it to the case of the standard hinge loss discriminator of [89]. For simplicity, let us consider a discriminator for the standard loss that can be written as the inner product $D_{\text{STN}}(x) = \phi(x)^\top \theta$, for some θ vector (this is updated only when we optimize with respect to the discriminator). In the case of our KDD loss we instead use simply $D_{\text{KDE}}(x) = \phi(x)$. Suppose that the discriminator is now given and we minimize the loss $\mathcal{L}_G$ with respect to the generator G. In the case of a first order optimization method, we obtain the updates for the parameters of the generator through the gradients of $\mathcal{L}_G$,

$$\frac{\partial \mathcal{L}_G}{\partial G} = \frac{\partial \mathcal{L}_G}{\partial \phi} \frac{\partial \phi}{\partial G}. \tag{6.10}$$

Since in both the standard hinge loss and our loss the term $\frac{\partial \phi}{\partial G}$ is the same, we can reduce the analysis to the study of $\frac{\partial \mathcal{L}_G}{\partial \phi}$. We obtain:

$$\frac{\partial \mathcal{L}_G^{\text{STN}}}{\partial \phi} = \theta \tag{6.11}$$

and

$$\frac{\partial \mathcal{L}_G^{KDD}}{\partial \phi} = \frac{1}{|S_g|} \sum_{x_g \in S_g} \frac{\partial \log \hat{p}_g^{\phi}(x_g)}{\partial \phi} - \frac{\log \hat{p}_r^{\phi}(x_g)}{\partial \phi}. \tag{6.12}$$

The formulas above show that in the case of the hinge loss the gradient update results in a constant shift, *i.e.*, an identical shift for all samples, whereas our KDD loss increases (resp. decreases) the likelihood of x_g under $\hat{p}_r^{\phi}$ (resp. $\hat{p}_g^{\phi}$). A illustration of this effect in 2D is shown in Fig. 6.1.

We also compare our KDD loss to the MMD loss proposed by [133]. Without loss of generality, for a given sample $x \sim p_1$ we compare each term $E_{y \sim p_2}[k(x, y)]$ in their work to its counterpart in ours $\log(E_{y \sim p_2}[k(x, y)])$, where $p_1, p_2 \in \{p_r, p_g\}$ and k is a kernel function. For the vMF kernel, we obtain

$$\text{KDD: } \sum_y \frac{k(x, y)}{\sum_v k(x, v)} \phi(y), \tag{6.13}$$

$$\text{MMD: } \sum_y k(x, y)\phi(y). \tag{6.14}$$

In both cases, the gradient is a weighted average of the samples $\phi(y)$. The key difference is that the Improved MMD loss has a local weighting, *i.e.*, it only depends on the current y, and the KDD loss has a global weighting.

Empirical Analysis of the KDD Loss

In Figure 6.1, we illustrate the difference between the Hinge and KDD losses. We consider two point clouds in 2D representing the real and fake push-forward distributions. In this example, the real point cloud is designed to have two Gaussian modes, while the fake one starts off with one uniformly sampled square mode. We first find the optimal linear classifier separating the two point clouds through gradient descent. The corresponding decision boundary is represented by the green line in Figure 6.1. We then optimize the features of the fake samples with respect to the Hinge loss and the KDD loss. In this example we do not normalize the feature mappings, since its main purpose is to prevent the discriminator from converging to degenerate solutions, where the space collapses. Thus, for visualization purposes, we work with 2D features. In this setting the vMF kernel is equivalent to a Gaussian kernel with $\sigma = 1$ for the KDE, *i.e.*, $\mathcal{K}(\phi, \xi) \propto \exp -\frac{|\phi - \xi|^2}{2}$.

The minimization of the Hinge loss simply results in translating the fake point cloud without changing its internal structure as shown in Figure 6.1b. In contrast, the KDD loss encourages the fake samples to head towards the closest real mode as shown in Figure 6.1c. For both of losses, the optimization was ran using SGD [12] for 200 iterations with a learning rate of 10. 1000 samples were used for both the real and fake

point clouds. Note that for a frozen Discriminator, updating the Generator using the Hinge loss can result in overshooting the real point cloud, since the translation vector is constant for all subsequent Generator updates. In fact, the optimum is to translate the fake point cloud to infinity. This makes the Generator update with respect to the Hinge loss less well-behaved than its KDD counterpart since the latter does not introduce such instability.

6.2.3 Class-Conditioning Extension

We also consider training generative models subject to class-conditioning. Let us denote with $y^{(i)}$ the label corresponding to the real image $x_r^{(i)}$. Now, we are interested in the approximation of the quantity $\log \frac{p_r(x,y)}{p_g(x,y)}$, which we can rewrite as

$$\log \frac{p_r(y|x)p_r(x)}{p_g(y|x)p_g(x)} = \log \frac{p_r(y|x)}{p_g(y|x)} + \log \frac{p_r(x)}{p_g(x)}. \tag{6.15}$$

The second term is exactly what we used in $\mathcal{L}_D(\phi, G)$ and $\mathcal{L}_G(\phi^*, G)$. Thus, we can focus on the conditional term $\log \frac{p_r(y|x)}{p_g(y|x)}$. By following [89], we assume the linear form

$$\log \frac{p_r(y|x)}{p_g(y|x)} = y^\top V D(x), \tag{6.16}$$

where V is a (learned) matrix that defines the embedding for the label y.

6.2.4 Regularization of the Feature Mapping

If ϕ maps many samples to the same feature, the discrimination task would become less effective. To avoid this scenario, we encourage φ, the feature mapping before the normalization layer, to be as "responsive" as possible to variations around samples of p_r and p_g by introducing the following additional *Jacobian regularization* term

$$\mathcal{L}_{\text{Jac}} = \frac{1}{|S_r|} \sum_{\substack{x \in S_r \bigcup S_g \\ \Delta x \sim \mathcal{U}(\mathbb{S}^{d-1})}} \left| \frac{|\varphi(x + \delta \Delta x) - \varphi(x)|_2}{\delta} - 1 \right|_1 \tag{6.17}$$

where $\delta > 0$ is a small scalar and Δx is a random unitary direction in image space. φ is defined so that $\phi = \varphi / |\varphi|_2$. This regularization term computes a finite difference approximation of the gradient of φ with respect to its input and projects it along the random direction Δx. It preserves as much as possible the volume in feature space, but only for the data in the image distribution. In addition, this regularization term prevents the magnification of the output gradient, which is typically associated with a high confidence, and would make the discriminator more susceptible to adversarial inputs. This is a stronger constraint compared to the classic gradient penalty [39], since we are implicitly requiring orthonormality for all the rows of the Jacobian, *i.e.*, $\nabla\varphi(x)\nabla\varphi(x)^\top = I_d$.

6.2.5 KDD GAN Formulation

Finally, we can put all the terms together and define the generator and discriminator losses via

$$\mathcal{L}_{G/D} = \gamma \mathcal{L}^{\text{KDD}}_{G/D} + \alpha \mathcal{L}^{\text{Hinge}}_{G/D} + \lambda_{\nabla} \mathcal{L}_{\text{Jac}}, \tag{6.18}$$

where γ, α and λ_{∇} live in $\mathbb{R}^{+} \times \{0, 1\} \times \{0, 1e\text{-}5\}$, and where KDD and Hinge refer to our KDD loss and the classic hinge loss used in BigGAN for both the generator and discriminator. The training with the lone hinge loss uses $\alpha = 1, \gamma = 0$; the training with the lone KDD loss uses $\alpha = 0, \gamma = 1$; the setting where $\alpha = 1, \gamma > 0$ is called `Joint` training.

6.3 Implementation

Training Details

We evaluate our models on three different datasets: CIFAR10 [63], Tiny ImageNet and ImageNet 64×64. The Tiny ImageNet [66] dataset is a subset of the ILSVRC-2012 ImageNet classification dataset [25] consisting of 200 object classes and 500 training images, 50 validation images and 50 test images per class. Unless specified otherwise, we use $\tau = 1$, $\delta = 1e\text{-}3$ and $\lambda_{\nabla} = 1e\text{-}5$. Experiments using data augmentations and the Jacobian regularization are denoted with **+DA** and **+JacD** respectively. All experiments were ran on at most two 2080Ti or one 3090Ti NVIDIA GPUs. Using KDD-GAN results in around 10% longer training times.

Architectures

The architecture used for our CIFAR10 experiments is the same one[1] used in the original BigGAN work by [14]. For both Tiny ImageNet and ImageNet 64×64, we use the modified SA-GAN [148] architecture adopted by [27] [2]. We do not use instance selection on CIFAR10 and Tiny ImageNet as we noticed it hurts performance on smaller datasets. For instance selection on ImageNet 64×64, we use a retention ratio of 50%. We choose to train BigGAN/SA-GAN rather than StyleGAN2-ADA for their simpler training scheme and their lesser reliance on regularization terms and implementation tricks. This allows use to isolate the contribution of our KDD loss without requiring a hyperparameter search for the rest of the moving pieces of the training.

[1] https://github.com/ajbrock/BigGAN-PyTorch/
[2] https://github.com/uoguelph-mlrg/instance_selection_for_gans/

Figure 6.2: Sample images generated using the Joint† model trained on ImageNet 64×64.

Evaluation Metrics

Throughout this paper, we evaluate our generative models using Fréchet Inception Distance (FID) [45], Inception Score (IS) [108], Density and Coverage [91]. These metrics are computed using the original *tensorflow* implementation. As in [27] the real moments used for the FID are computed using the entire dataset and not the filtered one. For FID and IS we use 50k generated samples, for Density and Coverage, we use 10k samples for both distributions and 5 nearest neighbors. Unless specified otherwise, the reported numbers are computed after 100k iterations for both CIFAR10 and Tiny ImageNet and after 500k iterations for ImageNet 64×64. The batch size used is 64 for Tiny ImageNet and CIFAR10 and 128 for ImageNet 64×64. The FID moments are computed on the training set for all datasets. We report the performance of the best model obtained during training.

Differentiable Augmentations

We use differentiable random brightness, saturation, contrast, translation and cut-out data augmentations proposed by [153]. For all our experiments, the loss is computed only on the non-augmented images. The augmented samples are only used for the Kernel Density Estimation. This is an important distinction from the work by [153].

6.4 Experiments

In this section we show the quantitative results obtained on CIFAR10, Tiny ImageNet and ImageNet 64×64. The best and second best values per metric are highlighted and underlined respectively. Generated samples from one of our best models are shown in Fig. 6.2.

Table 6.1: Comparison of the various BigGANs trained on CIFAR10. **UnCond** refers to the unconditional setting, while **NoProj** refers to removing the class-projection loss term in ProjGAN [89].

Experiments	τ	γ	**FID** ↓	**IS** ↑	**D** ↑	**C** ↑
Hinge	-	-	8.751	8.835	0.966	0.851
KDD	0.05	-	8.753	**9.233**	0.876	0.832
KDD	1.00	-	8.422	<u>9.155</u>	0.868	0.849
KDD	5.00	-	8.604	8.852	**0.970**	0.862
KDD + JacD	1.00	-	**7.237**	9.029	0.932	<u>0.867</u>
Joint	1.00	0.1	9.144	8.767	<u>0.969</u>	0.857
Joint	1.00	0.5	8.795	8.920	0.922	0.855
Joint	1.00	1.0	<u>7.932</u>	9.046	0.968	**0.868**
Joint	1.00	10.0	8.352	9.102	0.930	0.857
KDD + NoProj	0.05	-	13.668	8.274	0.722	0.711
Hinge (Uncond)	-	-	17.782	8.120	0.692	0.686
KDD (Uncond)	0.05	-	15.828	8.326	0.620	0.650
Joint (Uncond)	0.05	1.0	14.394	8.532	0.662	0.712

6.4.1 Ablation Results

In Table 6.1, we perform various ablations by training BigGAN [14] on CIFAR10 for 200k iterations each. The three main loss functions used are: the hinge loss [89], the KDD loss and the Joint loss. We study the effects of the parameters associated with the new losses. The first set of experiments studies the effect of the temperature τ used in the KDD loss. We observe that both high and low values of τ are problematic. When comparing $\tau = 0.05$ to $\tau = 5.00$, we observe a trade-off between Image Fidelity (FID) and diversity (IS). The value of τ determines the level of blurriness of the KDE. Additionally, we explore the effect of the Jacobian regularization. We use a coefficient of $\lambda_{\nabla} = 1e\text{-}5$. Our KDD GAN using $\tau = 1$ with and without the Jacobian regularization outperforms its BigGAN counterpart in both FID and IS. The performance gap is bigger when adding the Jacobian regularization.

The second set of experiments looks at the effect of γ during the joint training. We observe that all joint models improve on the baseline in terms of IS. This improvement correlates positively with γ except for $\gamma = 10$ where the IS stagnates. The best joint model ($\gamma = 1$) outperforms the baseline also in terms of FID. This highlights the benefit of using the KDD loss as a regularization term. Lastly, we train our models without the class-projection head proposed by [89] and/or without a conditional input for the

Table 6.2: KDD GAN kernel and dimensionality choice. We evaluate the impact of Kernel Choice and Feature Dimension on KDD GAN

Kernel	Feature Dimension	FID	IS
vMF	$K = 128$	8.384	8.887
IQ	$K = 128$	8.375	8.901
vMF	$K = 64$	8.842	8.885
vMF	$K = 128$	8.375	8.901
vMF	$K = 256$	9.050	9.058

generator. All models obtained with $\gamma > 0$ in the third block in Tab. 6.1 outperform the BigGAN baseline in the unconditional setting. This proves that training is not solely driven by the class-projection term in the conditional setting. The difference in performance between unconditional KDD model and the one that is only missing the projection head can be attributed to the slightly higher number of parameters that the latter has since it is still using the class label as input to the generator. We additionally examine the impact of the kernel choice and the dimension of the features on the KDD-GAN. The results are shown in Table 6.2 We compare the vMF kernel, which is equivalent to the RBF kernel due to the normalization used, to the IQ kernel [133]. We observe a similar performance level on CIFAR-10 for both kernel choices. Regarding the dimensionality K, we compare our default setting on CIFAR-10 ($K = 128$) to $K = 64$ and $K = 256$. Although increasing K slightly improves the IS, the best model overall remains the default one. We can conclude from both experiments that our KDD loss is not too sensitive to the choice of the kernel and dimension of the features as opposed to reported observations for models such as MMD-GAN [116].

6.4.2 Generative Learning on CIFAR10

In Table 6.3, we compare the performance of different variations of our KDD GAN with a BigGAN baseline and the numbers reported by [53] for a selection of their best models. The KDD GAN outperforms the BigGAN baseline for both IS and FID. Also it drastically improves its FID when using augmentations as described in Sec. 6.2.1. Augmentation $\times N$ means that an additional $N \times batchsize$ augmented images are used for the KDE anchor points. We observe that on CIFAR10, the amount of augmentations correlates positively with a significant improvement of the FID. In the case of the Jacobian regularization the results are mixed. It seems to improve the performance of the KDD model, but it also negatively impacts performance when used in combination with data augmentation. The Jacobian regularization may be too strict

Table 6.3: Quantitative results on CIFAR10. The values shown below are obtained after 100k iterations. We show the benefit of adding various augmentation factors for the KDD setting. We also explore the effect of the Jacobian regularization. * are numbers reported by [53].

Experiments	**FID** ↓	**IS** ↑	**D** ↑	**C** ↑
ContraGAN*	8.065	9.729	-	-
ContraGAN + DiffAug*	7.193	<u>9.996</u>	-	-
BigGAN + DiffAug*	**7.157**	9.775	-	-
BigGAN + CR*	<u>7.178</u>	**10.380**	-	-
Hinge loss	8.859	8.814	0.917	0.841
KDD	8.375	8.901	0.875	0.845
KDD + DA	7.089	9.250	0.893	0.860
KDD + DA ×3	<u>6.063</u>	9.280	<u>0.951</u>	<u>0.892</u>
KDD + DA ×7	**5.713**	**9.389**	**0.968**	**0.899**
KDD + JacD	7.944	8.959	0.895	0.847
KDD + JacD + DA×7	6.713	<u>9.333</u>	0.9000	0.875

a requirement, as the dimension K of the gradient of ϕ is smaller than the dimension d of the images, and perhaps a more flexible loss term could work better.

6.4.3 Generative Learning on ImageNet

Tiny ImageNet

Table 6.4 shows the performance of our models on Tiny ImageNet compared to the SA-GAN baseline and the best models reported by [53]. The KDD GAN outperforms the baseline for all settings. On one hand, similarly to CIFAR10, using additional augmented images for the KDE results in a significant boost in performance. Indeed the KDD GAN with DA ×3 outperforms ContraGAN in terms of FID and IS. On the other hand, the additional Jacobian regularization is not helpful. The only exception being the joint training ($\gamma = 0.5$) without data augmentation and the joint training with $\gamma = 1$ and data augmentation where the Jacobian regularization introduces a slight performance boost. Note that the ContraGAN+Diff.Aug. numbers reported by [53] were obtained twice as many iterations as the rest of the models (ContraGAN and our experiments), putting it at an advantage.

Table 6.4: Quantitative results on Tiny ImageNet. We compare the baseline to both the KDD and joint trainings. We also explore the effect of adding the Jacobian regularization on D and show the effect of using more augmentations for the density estimation. * are numbers reported by [53].

Experiments	γ	**FID** $\downarrow$	**IS** $\uparrow$	**D** $\uparrow$	**C** $\uparrow$
ContraGAN*	-	27.027	13.494	-	-
+ DiffAugment*	-	**15.755**	**17.303**	-	-
Hinge loss	-	29.525	11.048	0.520	0.516
KDD	-	24.022	13.204	0.658	0.613
KDD+DA	-	<u>20.204</u>	<u>14.100</u>	<u>0.673</u>	<u>0.663</u>
KDD+DA $\times 3$	-	**18.261**	**14.943**	**0.716**	**0.683**
KDD+JacD	-	25.504	13.215	0.597	0.595
KDD+JacD+DA	-	20.717	13.787	0.630	0.645
Joint	1	25.709	13.124	0.595	0.582
Joint+DA	1	22.854	13.421	0.591	0.613
Joint+JacD	1	26.369	13.169	0.582	0.582
Joint+JacD+DA	1	21.512	13.728	0.639	0.627
Joint	0.5	24.341	13.337	0.626	0.614
Joint+DA	0.5	23.357	12.918	0.619	0.621
Joint+JacD	0.5	23.854	13.251	0.651	0.617
Joint+JacD+DA	0.5	23.928	13.059	0.575	0.594

ImageNet 64×64

Table 6.5 shows our experimental results on ImageNet 64×64. We compare our models to the SA-GAN baseline and the numbers reported by [27] and [154]. For all our trained models, we use instance selection [27] with a retention ratio of 50%.

We observe that the baseline outperforms our KDD GAN even with additional augmentations and regularization. It is also note-worthy that in this setting, although a small amount of data augmentation seems to help, adding more is not necessarily beneficial. The high level of diversity in ImageNet both in terms of number of classes and samples might be limiting the effectiveness of our density estimation given the relatively small batch size used. Nevertheless, all joint training models outperform the hinge-based models in terms of IS and most outperform our SA-GAN baseline in terms of FID. Interestingly, the best model is the Joint† model where $\hat{p}_r$ is estimated using features computed during the last discriminator step. This suggests that using a memory bank for the features might be a promising extension of this work.

Table 6.5: Quantitative results on ImageNet 64×64. We explore the use of augmentation, Jacobian regularization and Joint training. † refers to a setting where the feature $\phi(x_r)$ were computed using the weights from the previous discriminator update step. * are numbers reported by [27].

Experiments	γ	**FID** ↓	**IS** ↑	**D** ↑	**C** ↑
SA-GAN+IS@50%*	-	9.63	31.04	1.07	0.88
FQ-BigGAN*	-	9.67	25.96	-	-
Hinge loss	-	10.452	32.869	1.034	0.877
KDD	-	12.570	31.404	0.953	0.850
KDD+DA	-	12.367	31.069	0.954	0.861
KDD+DA ×3	-	14.680	27.949	0.928	0.810
KDD+JacD	-	12.651	31.188	0.938	0.850
KDD+JacD+DA	-	79.790	10.603	0.376	0.139
Joint	1	11.387	32.471	0.991	0.872
Joint+DA	1	10.385	33.753	1.048	0.880
Joint+JacD	1	10.320	34.296	1.010	0.868
Joint+JacD+DA	1	9.702	34.619	1.062	0.892
Joint	0.5	10.544	33.447	1.017	0.879
Joint †	0.5	**9.450**	**35.648**	1.070	**0.897**
Joint+DA	0.5	10.111	33.494	1.048	0.891
Joint+JacD	0.5	10.242	35.120	**1.072**	0.891
Joint+JacD+DA	0.5	10.010	34.074	1.053	0.889

6.5 Examples of Generated Images

We show non-cherry picked images generated by our Hinge loss baseline and our best model per dataset in Figures 6.3 to 6.9. The truncation trick for sampling [14] was not used. In all figures, each row represents a different class starting with the first class in the top row down to the last class in the bottom row.

6.6 Discussion

One of the main challenges in the use of KDD GAN is to ensure that the anchor points for the KDE are representative for the evaluation points. In our experiments between Tiny ImageNet and ImageNet 64×64, we observe that the performance of KDD GAN is sensitive to the anchor points set size, the number of augmentations, and the complexity of the dataset seems to play a role as well. Also, with large datasets the impact of samples at the tails of the distribution on the KDE approximation is

(a) Hinge loss (b) KDD + Aug ×7

Figure 6.3: Samples generated using the Hinge loss model and the KDD + Aug ×7 model trained on CIFAR10 (one class per row).

unclear. In general, it might be necessary to design better sampling strategies for the anchor points used for the KDE estimation: Some options are using a memory bank or sampling using k-NN. Another direction to evaluate is the role of the class projection in the training. We chose to separate the category aspect from the unlabeled problem not only because it would make KDD GAN suitable for unsupervised learning, but also because it would not require large minibatches as the current KDE completely ignores the class labels. It would be interesting to evaluate the performance in the case where the loss with class labels is entirely based on the KDE. Finally, as mentioned in the introduction, KDD GAN can be combined with other techniques and regularization methods that are known to improve the performance of the GAN training, such as Consistency Regularization of [149] and Differentiable Augmentation of [153]. We leave these investigations to future work.

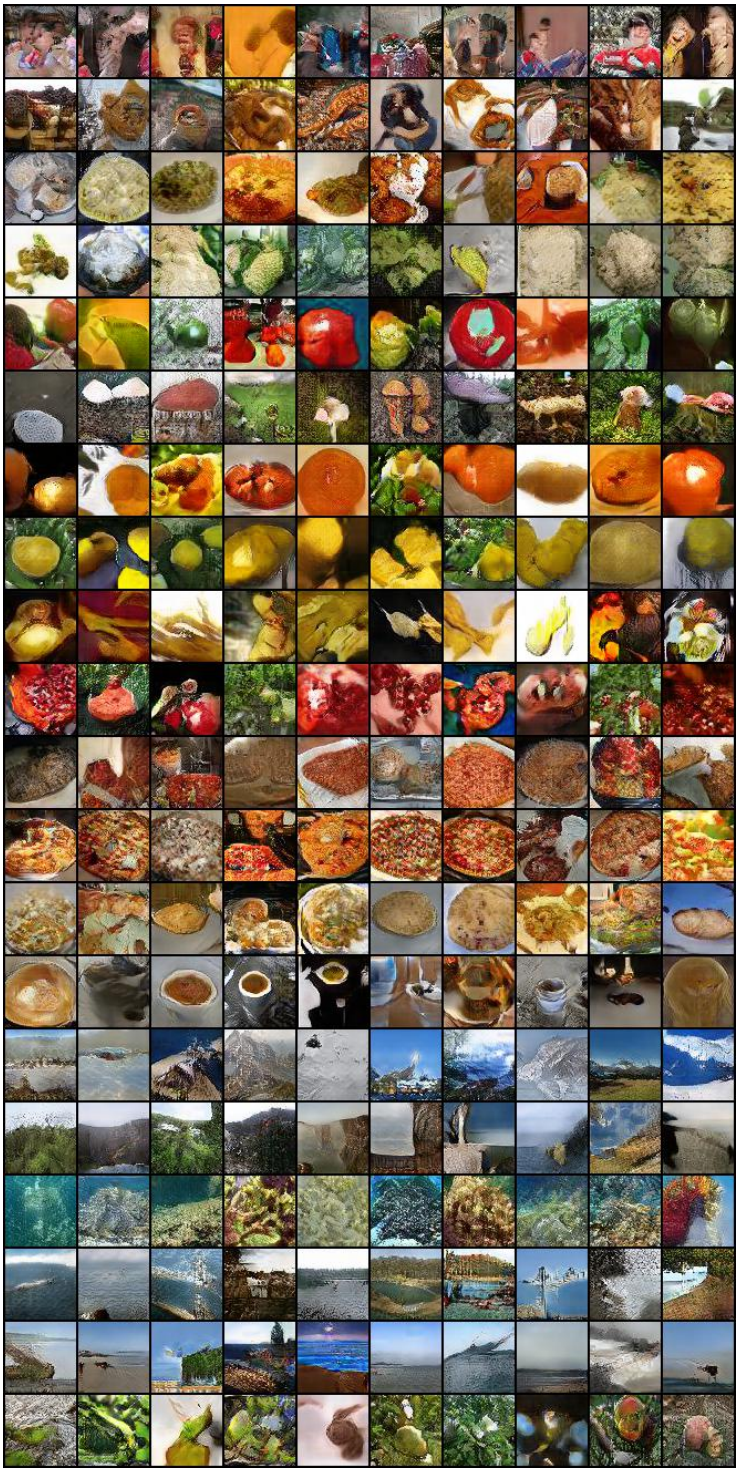


Figure 6.4: Samples generated using the Hinge loss model trained on Tiny ImageNet for the classes 181-200 (one class per row).

Figure 6.5: Samples generated using the KDD+Aug $\times 3$ model trained on Tiny ImageNet for the classes 181-200 (one class per row).

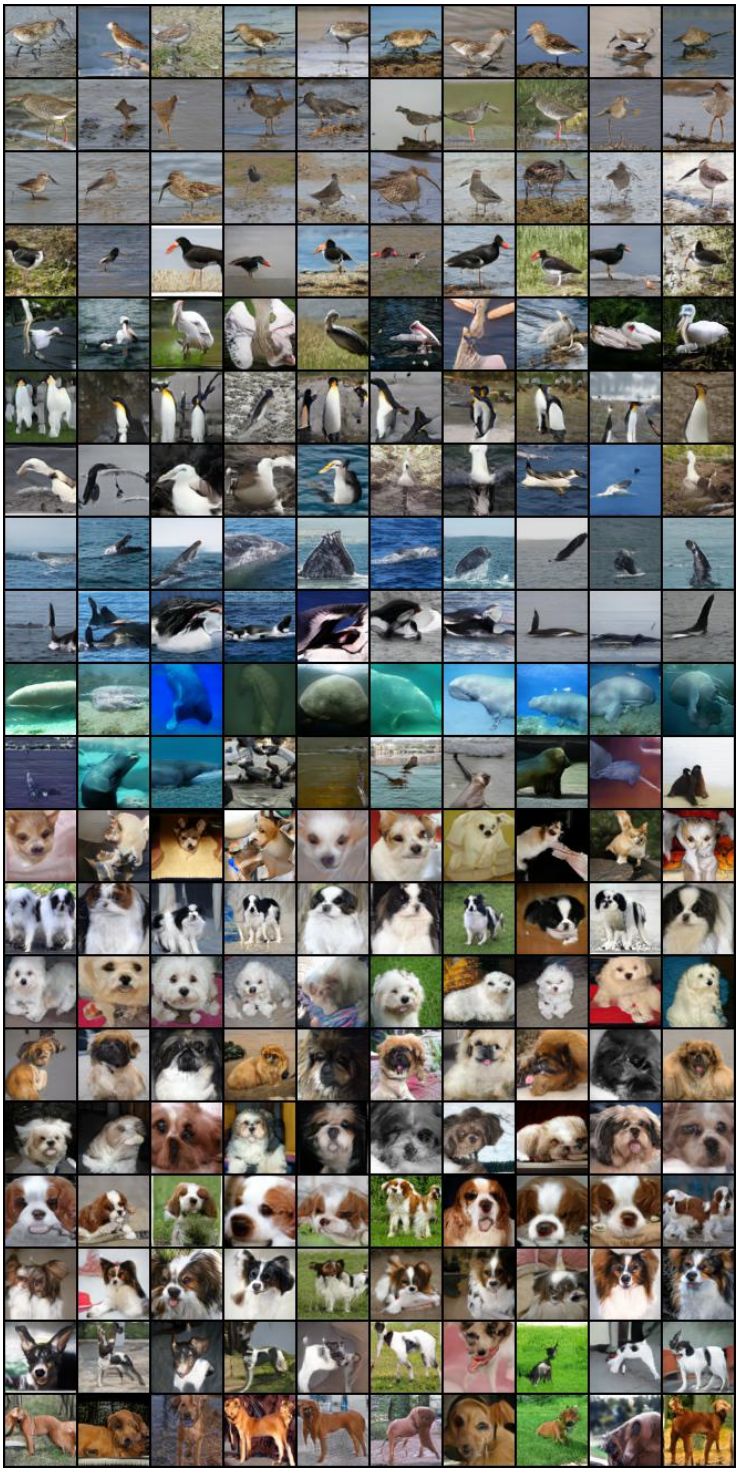


Figure 6.6: Samples generated using Hinge loss model trained on ImageNet 64×64 for the classes 141-160 (one class per row).

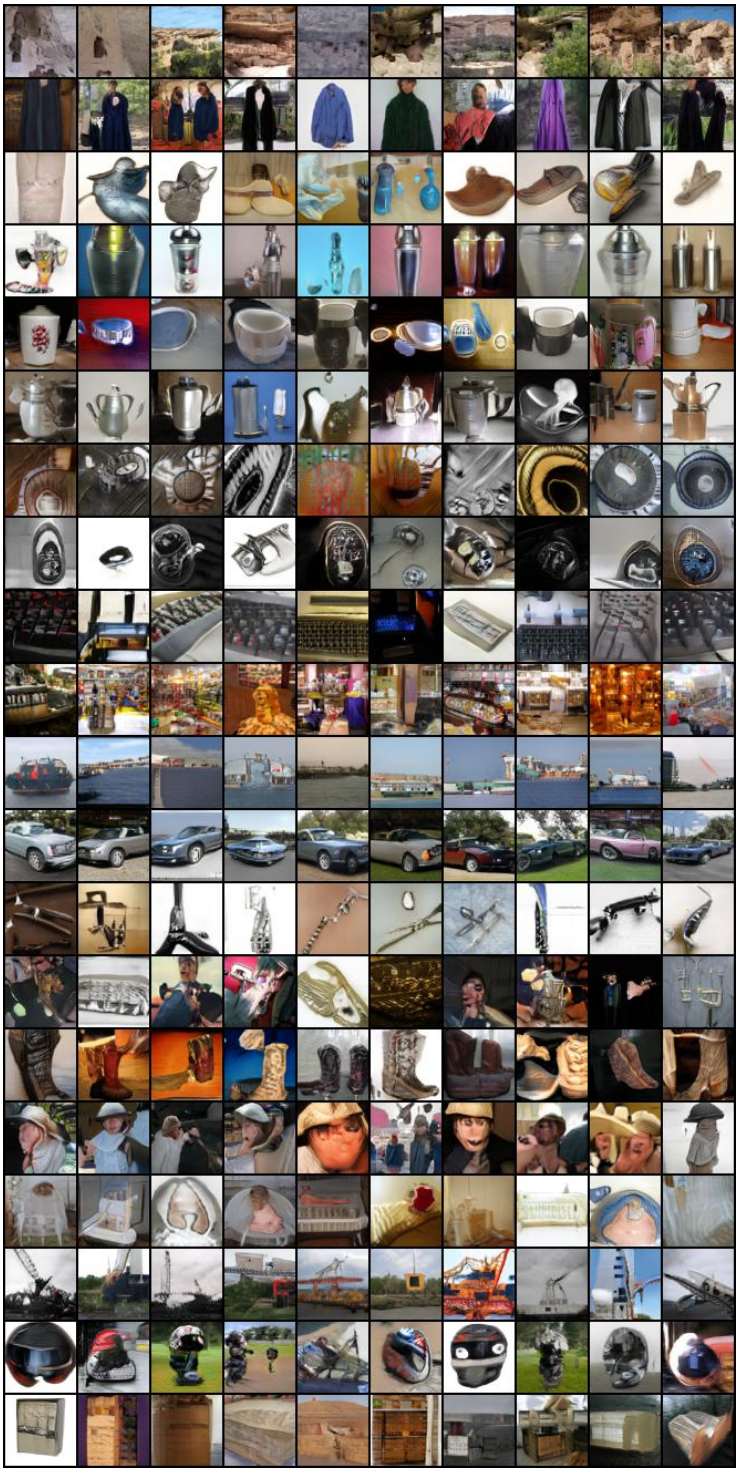


Figure 6.7: Samples generated using Hinge loss model trained on ImageNet 64×64 for the classes 501-520 (one class per row).

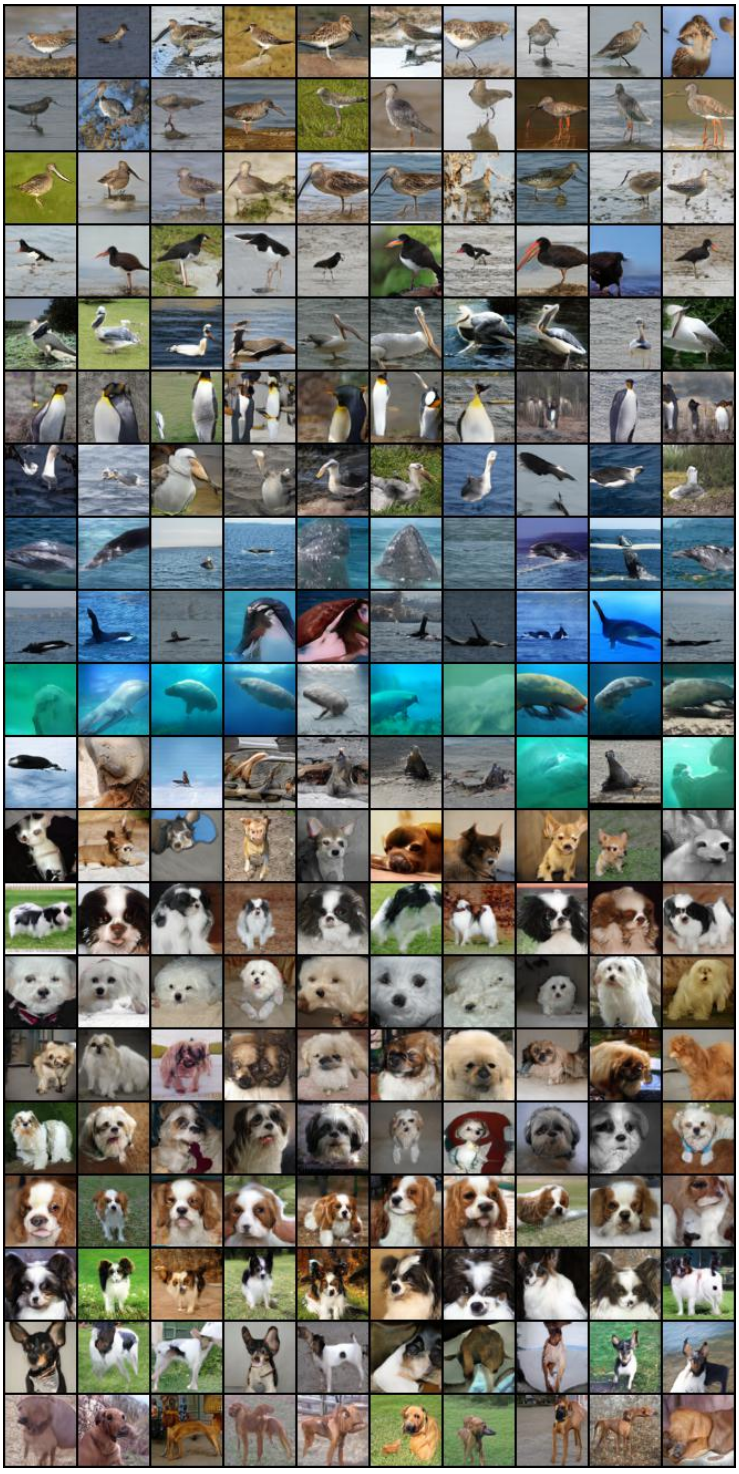


Figure 6.8: Samples generated using Joint† model trained on ImageNet 64×64 for the classes 141-160 (one class per row).

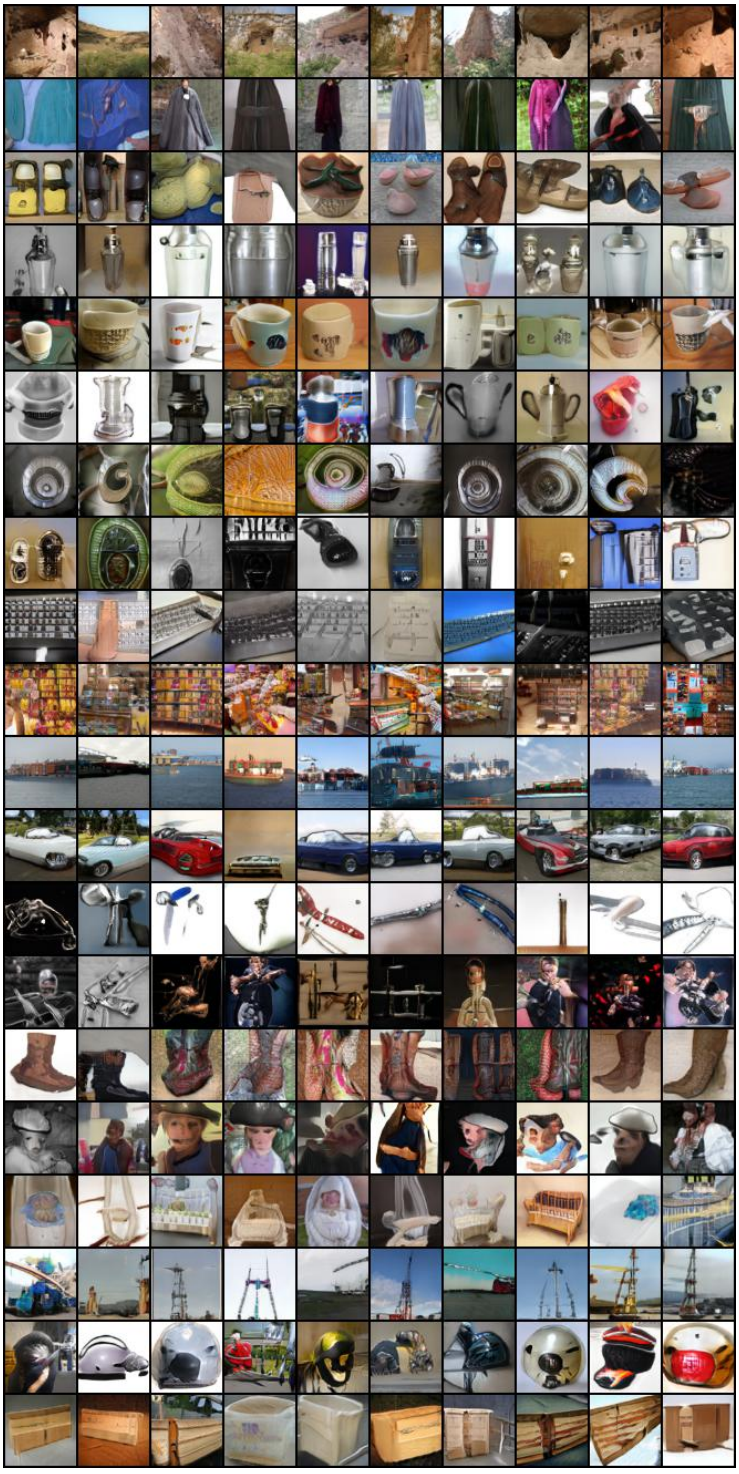


Figure 6.9: Samples generated using Joint† model trained on ImageNet 64×64 for the classes 501-520 (one class per row).

Chapter 7

Conclusions

In this thesis, we explored the challenging problem of unsupervised object segmentation. We demonstrated the feasibility of training both *generative* models, which are capable of generating segmented objects, and *segmenter* models, which can identify and segment salient objects from real images. Notably, the methods we introduced can be trained solely using collections of images, eliminating the need for any form of manual annotations – such as object labels, bounding boxes, landmarks, or the use of pre-trained supervised object detectors and classifiers.

We identified a powerful signal that can drive object segmentation: objects within a scene move *coherently*, meaning all parts of an object move together. This inherent property of physical objects can be embedded as a learning rule to steer deep neural networks towards segmenting objects that *can* be moved. We demonstrated the applicability of this principle to collections of *static*, *still* images, in which motion is not directly observable. Through generative modeling and inpainting networks, we devised methods to simulate realistic object displacement via local shifts. In Chapter 3, we introduced a novel method for generating a layered scene representation, composed of background and foreground layers, along with a corresponding mask for the foreground object. By enforcing the constraint that the scene must remain realistic under a local shift of the foreground, we achieved a meaningful separation between the foreground and background. To improve our model's capability to segment real images, we outlined in Chapter 4 how to streamline these models and use the synthetic data to train a competitive segmenter that performs well on real images. In Chapter 5 we departed from relying on a generative model altogether, by adopting modern self-supervised models as the feature backbone and as the inpainter. Inpainting the background not only allows for local shifts of the predicted objects within real images but also generates artifacts indicative of incorrect segmentation. We showed how to utilize it as an additional signal driving the segmentation. Our methods heavily rely on adversarial training to ensure the generated or modified scenes remain realistic. In an effort to

improve this training process, we introduce in Chapter 6 a new GAN training method with a novel loss function. This function utilizes Kernel Density Discrimination to measure the statistical divergence between the kernel density estimates of real and fake data distributions within the discriminator's feature space. This results in improved training gradients, encouraging the generator to seek missing distribution modes.

Unsupervised object segmentation has gained increasing attention in recent years. This line of work is significant for reducing the reliance on costly, expertise-demanding annotations and enhancing the capabilities of current object segmentation systems. Perhaps more significantly, this approach represents a step towards the development of more generalized artificial intelligence agents, capable of learning by simply observing the world. Depending on the use case, there are several possible directions to continue our work.

Semi-Supervised and Weakly Supervised Learning. While researching purely unsupervised methods is significant beyond cost-effectiveness, in many use cases it is possible to obtain at least some annotations with limited effort, such as using less precise bounding boxes or fewer instance labels. Incorporating our principles for unsupervised segmentation within a semi-supervised or weakly supervised framework could further improve practical systems that could be deployed in real-world scenarios.

Extending Self-Supervised Learning. As shown in this thesis, desirable feature properties for object segmentation emerge from modern transformer-based self-supervised methods. Further research investigates the improvement of such properties [24]. Merging the typical SSL learning objectives with object-aware objectives presented in this thesis could lead to a richer feature representation.

Addressing Dataset Imposed Limitations and Object Ambiguity. Most commonly used image datasets, including ImageNet [25], are object-centric, which benefits many self-supervised models by allowing them to impose feature invariance under geometric transformations. While our methods do not rely on explicit labels, their effectiveness might still depend on such curated datasets. Exploring extensions for multi-object discovery, possibly through object-centric slot attention models [112], could offer a way to achieve precise yet distinct object masks

Beyond Generative Adversarial Networks. The adversarial learning framework is desirable in our case for end-to-end learning of realistic scenes. Diffusion models have recently surpassed GANs in generating high-quality images, but are not directly applicable to our methods to provide the signal guiding the realism of a scene. However, there is a recent line of work that utilizes pre-trained diffusion models to obtain a

guiding signal towards a constrained image generation, e.g. a 3D representation [102]. Further research in this direction is needed, since these methods currently can suffer from mode collapse and are strongly dependent on text conditioning.

Bibliography

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34 (11):2274–2282, November 2012.

[2] Shir Amir, Yossi Gandelsman, Shai Bagon, and Tali Dekel. Deep vit features as dense visual descriptors. *ArXiv preprint*, abs/2112.05814, 2021.

[3] Relja Arandjelović and Andrew Zisserman. Object discovery with a copy-pasting gan. *ArXiv preprint*, abs/1905.11369, 2019.

[4] Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR, 2017.

[5] Amir Bar, Xin Wang, Vadim Kantorov, Colorado J. Reed, Roei Herzig, Gal Chechik, Anna Rohrbach, Trevor Darrell, and Amir Globerson. Detreg: Unsupervised pretraining with region priors for object detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 14585–14595. IEEE, 2022.

[6] H. B. Barlow. Unsupervised learning. *Neural Computation*, 1:295–311, 1989.

[7] Jonathan T Barron and Ben Poole. The fast bilateral solver. In *European Conference on Computer Vision*, pages 617–632. Springer.

[8] Yaniv Benny and Lior Wolf. Onegan: Simultaneous unsupervised learning of conditional image generation, foreground segmentation, and fine-grained clustering. In *European Conference on Computer Vision*, pages 514–530. Springer, 2020.

[9] David Berthelot, Thomas Schumm, and Luke Metz. BEGAN: Boundary equilibrium generative adversarial networks. *ArXiv preprint*, abs/1703.10717, 2017.

[10] Adam Bielski and Paolo Favaro. Emergence of object segmentation in perturbed generative models. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 7254–7264, 2019.

[11] Adam Bielski and Paolo Favaro. MOVE: Unsupervised movable object segmentation and detection. *Advances in Neural Information Processing Systems*, 35: 33371–33386, 2022.

[12] Léon Bottou. Online algorithms and stochastic approximations. *Online learning and neural networks*, 1998.

[13] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[14] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *7^{th} International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[15] Christopher P Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. MONET: Unsupervised scene decomposition and representation. *ArXiv preprint*, abs/1901.11390, 2019.

[16] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[17] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 9630–9640. IEEE, 2021.

[18] Mickaël Chen, Thierry Artières, and Ludovic Denoyer. Unsupervised object segmentation by redrawing. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on*

Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, pages 12705–12716, 2019.

[19] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR, 2020.

[20] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 9620–9629. IEEE, 2021.

[21] Bowen Cheng, Alexander G. Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 17864–17875, 2021.

[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 *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 1201–1210. IEEE Computer Society, 2015.

[23] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24:603–619, 2002.

[24] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *The Twelfth International Conference on Learning Representations*, 2024.

[25] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 248–255. IEEE Computer Society, 2009.

[26] Thomas Deselaers, Bogdan Alexe, and Vittorio Ferrari. Localizing objects while learning their appearance. In *European conference on computer vision*, pages 452–466. Springer, 2010.

[27] Terrance DeVries, Michal Drozdzal, and Graham W. Taylor. Instance selection for gans. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[28] 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 *9^{th} International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

[29] Aysegul Dundar, Karan Sapra, Guilin Liu, Andrew Tao, and Bryan Catanzaro. Panoptic-based image synthesis. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 8067–8076. IEEE, 2020.

[30] S. M. Ali Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, Koray Kavukcuoglu, and Geoffrey E. Hinton. Attend, infer, repeat: Fast scene understanding with generative models. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3225–3233, 2016.

[31] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html, .

[32] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html, .

[33] Zoubin Ghahramani. *Unsupervised Learning*, pages 72–112. Springer Berlin Heidelberg, 2004.

[34] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *6^{th} International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[35] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680, 2014.

[36] Klaus Greff, Antti Rasmus, Mathias Berglund, Tele Hotloo Hao, Harri Valpola, and Jürgen Schmidhuber. Tagger: Deep unsupervised perceptual grouping. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4484–4492, 2016.

[37] Klaus Greff, Sjoerd van Steenkiste, and Jürgen Schmidhuber. Neural expectation maximization. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6691–6701, 2017.

[38] Klaus Greff, Raphaël Lopez Kaufman, Rishabh Kabra, Nick Watters, Christopher Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2424–2433. PMLR, 2019.

[39] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5767–5777, 2017.

[40] Mark Hamilton, Zhoutong Zhang, Bharath Hariharan, Noah Snavely, and William T. Freeman. Unsupervised semantic segmentation by distilling feature correspondences. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022.

[41] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016.

[42] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2980–2988. IEEE Computer Society, 2017.

[43] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum contrast for unsupervised visual representation learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 9726–9735. IEEE, 2020.

[44] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross B. Girshick. Masked autoencoders are scalable vision learners. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 15979–15988. IEEE, 2022.

[45] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6626–6637, 2017.

[46] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[47] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32^{nd} International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org, 2015.

[48] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 5967–5976. IEEE Computer Society, 2017.

[49] Jongheon Jeong and Jinwoo Shin. Training gans with stronger augmentations via contrastive discriminator. In *9^{th} International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

[50] Xu Ji, Andrea Vedaldi, and João F. Henriques. Invariant information clustering for unsupervised image classification and segmentation. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 9864–9873. IEEE, 2019.

[51] Bowen Jiang, Lihe Zhang, Huchuan Lu, Chuan Yang, and Ming-Hsuan Yang. Saliency detection via absorbing markov chain. In *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, pages 1665–1672. IEEE Computer Society, 2013.

[52] 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, Portland, OR, USA, June 23-28, 2013*, pages 2083–2090. IEEE Computer Society, 2013.

[53] Minguk Kang and Jaesik Park. Contragan: Contrastive learning for conditional image generation. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[54] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *6^{th} International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[55] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 4401–4410. Computer Vision Foundation / IEEE, 2019.

[56] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[57] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 8107–8116. IEEE, 2020.

[58] Isinsu Katircioglu, Helge Rhodin, Victor Constantin, Jörg Spörri, Mathieu Salzmann, and Pascal Fua. Self-supervised human detection and segmentation via background inpainting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021.

[59] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[60] Gunhee Kim and Antonio Torralba. Unsupervised detection of regions of interest using iterative link analysis. In Yoshua Bengio, Dale Schuurmans, John D. Lafferty, Christopher K. I. Williams, and Aron Culotta, editors, *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada*, pages 961–969. Curran Associates, Inc., 2009.

[61] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[62] Ivan Kobyzev, Simon Prince, and Marcus Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[63] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[64] Orest Kupyn, Volodymyr Budzan, Mykola Mykhailych, Dmytro Mishkin, and Jiri Matas. Deblurgan: Blind motion deblurring using conditional adversarial networks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 8183–8192. IEEE Computer Society, 2018.

[65] Hanock Kwak and Byoung-Tak Zhang. Generating images part by part with composite generative adversarial networks. *ArXiv preprint*, abs/1607.05387, 2016.

[66] Ya Le and X. Yang. Tiny imagenet visual recognition challenge. 2015.

[67] 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.

[68] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew P. Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 105–114. IEEE Computer Society, 2017.

[69] Abdelhak Lemkhenter, Adam Bielski, Alp Eren Sari, and Paolo Favaro. Generative adversarial learning via kernel density discrimination. *arXiv preprint arXiv:2107.06197*, 2021.

[70] Bowen Li, Xiaojuan Qi, Thomas Lukasiewicz, and Philip H. S. Torr. Manigan: Text-guided image manipulation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 7877–7886. IEEE, 2020.

[71] Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. MMD GAN: towards deeper understanding of moment matching network. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 2203–2213, 2017.

[72] Guanbin Li and Yizhou Yu. Visual saliency based on multiscale deep features. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 5455–5463. IEEE Computer Society, 2015.

[73] Nianyi Li, Bilin Sun, and Jingyi Yu. A weighted sparse coding framework for saliency detection. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 5216–5223. IEEE Computer Society, 2015.

[74] Xiaohui Li, Huchuan Lu, Lihe Zhang, Xiang Ruan, and Ming-Hsuan Yang. Saliency detection via dense and sparse reconstruction. In *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, pages 2976–2983. IEEE Computer Society, 2013.

[75] Chieh Hubert Lin, Yen-Chi Cheng, Hsin-Ying Lee, Sergey Tulyakov, and Ming-Hsuan Yang. InfinityGAN: Towards infinite-pixel image synthesis. In *International Conference on Learning Representations*, 2022.

[76] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2014.

[77] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollar, and Larry Zitnick. Microsoft coco: Common objects in context. In *ECCV*. European Conference on Computer Vision, September 2014.

[78] Zinan Lin, Ashish Khetan, Giulia C. Fanti, and Sewoong Oh. Pacgan: The power of two samples in generative adversarial networks. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 1505–1514, 2018.

[79] Rui Liu, Yixiao Ge, Ching Lam Choi, Xiaogang Wang, and Hongsheng Li. Divco: Diverse conditional image synthesis via contrastive generative adversarial network. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 16377–16386. Computer Vision Foundation / IEEE, 2021.

[80] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[81] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 3431–3440. IEEE Computer Society, 2015.

[82] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2813–2821. IEEE Computer Society, 2017.

[83] Francisco Massa and Ross Girshick. maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch. https://github.com/facebookresearch/maskrcnn-benchmark, 2018. Accessed: 05/20/2019.

[84] 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.

[85] Luke Melas-Kyriazi, Christian Rupprecht, Iro Laina, and Andrea Vedaldi. Finding an unsupervised image segmenter in each of your deep generative models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022.

[86] Sachit Menon, Alexandru Damian, Shijia Hu, Nikhil Ravi, and Cynthia Rudin. PULSE: self-supervised photo upsampling via latent space exploration of generative models. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 2434–2442. IEEE, 2020.

[87] Lars M. Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3478–3487. PMLR, 2018.

[88] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[89] Takeru Miyato and Masanori Koyama. Cgans with projection discriminator. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[90] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[91] Muhammad Ferjad Naeem, Seong Joon Oh, Youngjung Uh, Yunjey Choi, and Jaejun Yoo. Reliable fidelity and diversity metrics for generative models. In *Proceedings of the 37th International Conference on Machine Learning, ICML*

2020, 13-18 July 2020, Virtual Event, volume 119 of *Proceedings of Machine Learning Research*, pages 7176–7185. PMLR, 2020.

[92] 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 via self-supervision. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 204–214, 2019.

[93] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729. IEEE, 2008.

[94] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 1520–1528. IEEE Computer Society, 2015.

[95] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European conference on computer vision*, pages 69–84. Springer, 2016.

[96] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. F-gan: Training generative neural samplers using variational divergence minimization. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 271–279, 2016.

[97] Pavel Ostyakov, Roman Suvorov, Elizaveta Logacheva, Oleg Khomenko, and Sergey I. Nikolenko. SEIGAN: towards compositional image generation by simultaneously learning to segment, enhance, and inpaint. *ArXiv preprint*, abs/1811.07630, 2018.

[98] Yassine Ouali, Céline Hudelot, and Myriam Tami. Autoregressive unsupervised image segmentation. In *European Conference on Computer Vision*, pages 142–158. Springer, 2020.

[99] Xingang Pan, Xiaohang Zhan, Bo Dai, Dahua Lin, Chen Change Loy, and Ping Luo. Exploiting deep generative prior for versatile image restoration and

manipulation. In *European Conference on Computer Vision*, pages 262–277. Springer, 2020.

[100] Taesung Park, Alexei A Efros, Richard Zhang, and Jun-Yan Zhu. Contrastive learning for unpaired image-to-image translation. In *European Conference on Computer Vision*, pages 319–345. Springer, 2020.

[101] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035, 2019.

[102] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *The Eleventh International Conference on Learning Representations*, 2023.

[103] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8821–8831. PMLR, 2021.

[104] Scott E. Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1060–1069. JMLR.org, 2016.

[105] Tal Remez, Jonathan Huang, and Matthew Brown. Learning to segment via cut-and-paste. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[106] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In Corinna

Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 91–99, 2015.

[107] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[108] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2226–2234, 2016.

[109] Axel Sauer, Kashyap Chitta, Jens Müller, and Andreas Geiger. Projected gans converge faster. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 17480–17492, 2021.

[110] Pedro Savarese, Sunnie S. Y. Kim, Michael Maire, Greg Shakhnarovich, and David McAllester. Information-theoretic segmentation by inpainting error maximization. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 4029–4039. Computer Vision Foundation / IEEE, 2021.

[111] Edgar Schönfeld, Bernt Schiele, and Anna Khoreva. A u-net based discriminator for generative adversarial networks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 8204–8213. IEEE, 2020.

[112] Maximilian Seitzer, Max Horn, Andrii Zadaianchuk, Dominik Zietlow, Tianjun Xiao, Carl-Johann Simon-Gabriel, Tong He, Zheng Zhang, Bernhard Schölkopf, Thomas Brox, and Francesco Locatello. Bridging the gap to real-world object-centric learning. In *The Eleventh International Conference on Learning Representations*, 2023.

[113] 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, 2015.

[114] Gyungin Shin, Samuel Albanie, and Weidi Xie. Unsupervised salient object detection with spectral cluster voting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2022, New Orleans, LA, USA, June 19-20, 2022*, pages 3970–3979. IEEE, 2022.

[115] 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. November 2021.

[116] Mathieu Sinn and Ambrish Rawat. Non-parametric estimation of jensen-shannon divergence in generative adversarial network training. In Amos J. Storkey and Fernando Pérez-Cruz, editors, *International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain*, volume 84 of *Proceedings of Machine Learning Research*, pages 642–651. PMLR, 2018.

[117] Parthipan Siva, Chris Russell, Tao Xiang, and Lourdes Agapito. Looking beyond the image: Unsupervised learning for object saliency and detection. In *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*, pages 3238–3245. IEEE Computer Society, 2013.

[118] Elizabeth S Spelke. Principles of object perception. *Cognitive science*, 14(1): 29–56, 1990.

[119] Elizabeth S Spelke. *What Babies Know: Core Knowledge and Composition Volume 1*, volume 1. Oxford University Press, 2022.

[120] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U. Gutmann, and Charles Sutton. VEEGAN: reducing mode collapse in gans using implicit variational learning. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 3308–3318, 2017.

[121] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *European conference on computer vision*, pages 776–794. Springer, 2020.

[122] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[123] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Deep image prior. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 9446–9454. IEEE Computer Society, 2018.

[124] Sjoerd van Steenkiste, Karol Kurach, Jürgen Schmidhuber, and Sylvain Gelly. Investigating object compositionality in generative adversarial networks. *Neural Networks*, 130:309–325, 2020.

[125] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[126] Huy V. Vo, Francis R. Bach, Minsu Cho, Kai Han, Yann LeCun, Patrick Pérez, and Jean Ponce. Unsupervised image matching and object discovery as optimization. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 8287–8296. Computer Vision Foundation / IEEE, 2019.

[127] Huy V Vo, Patrick Pérez, and Jean Ponce. Toward unsupervised, multi-object discovery in large-scale image collections. In *European Conference on Computer Vision*, pages 779–795. Springer, 2020.

[128] Huy V. Vo, Elena Sizikova, Cordelia Schmid, Patrick Pérez, and Jean Ponce. Large-scale unsupervised object discovery. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 16764–16778, 2021.

[129] Andrey Voynov, Stanislav Morozov, and Artem Babenko. Object segmentation without labels with large-scale generative models. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 10596–10606. PMLR, 2021.

[130] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[131] 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 *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 3796–3805. IEEE Computer Society, 2017.

[132] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 9929–9939. PMLR, 2020.

[133] Wei Wang, Yuan Sun, and Saman K. Halgamuge. Improving MMD-GAN training with repulsive loss function. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[134] Xinlong Wang, Rufeng Zhang, Chunhua Shen, Tao Kong, and Lei Li. Dense contrastive learning for self-supervised visual pre-training. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 3024–3033. Computer Vision Foundation / IEEE, 2021.

[135] Xinlong Wang, Zhiding Yu, Shalini De Mello, Jan Kautz, Anima Anandkumar, Chunhua Shen, and Jose M Alvarez. FreeSOLO: Learning to segment objects without annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14176–14186, 2022.

[136] Yangtao Wang, Xi Shen, Shell Xu Hu, Yuan Yuan, James L Crowley, and Dominique Vaufreydaz. Self-supervised transformers for unsupervised object discovery using normalized cut. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14543–14553, 2022.

[137] Xiu-Shen Wei, Chen-Lin Zhang, Jianxin Wu, Chunhua Shen, and Zhi-Hua Zhou. Unsupervised object discovery and co-localization by deep descriptor transformation. *Pattern Recognition*, 88:113–126, 2019.

[138] Qiong Yan, Li Xu, Jianping Shi, and Jiaya Jia. Hierarchical saliency detection. In *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*, pages 1155–1162. IEEE Computer Society, 2013.

[139] 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, Portland, OR, USA, June 23-28, 2013*, pages 3166–3173. IEEE Computer Society, 2013.

[140] Jianwei Yang, Anitha Kannan, Dhruv Batra, and Devi Parikh. LR-GAN: layered recursive generative adversarial networks for image generation. In *5^{th} International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[141] Yanchao Yang, Antonio Loquercio, Davide Scaramuzza, and Stefano Soatto. Unsupervised moving object detection via contextual information separation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 879–888. Computer Vision Foundation / IEEE, 2019.

[142] Zhaoyuan Yin, Pichao Wang, Fan Wang, Xianzhe Xu, Hanling Zhang, Hao Li, and Rong Jin. Transfgu: A top-down approach to fine-grained unsupervised semantic segmentation. In *European conference on computer vision*, pages 73–89. Springer, 2022.

[143] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *ArXiv preprint*, abs/1506.03365, 2015.

[144] Ning Yu, Ke Li, Peng Zhou, Jitendra Malik, Larry Davis, and Mario Fritz. Inclusive gan: Improving data and minority coverage in generative models. In *European Conference on Computer Vision*, pages 377–393. Springer, 2020.

[145] Ning Yu, Guilin Liu, Aysegul Dundar, Andrew Tao, Bryan Catanzaro, Larry Davis, and Mario Fritz. Dual contrastive loss and attention for gans. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 6711–6722. IEEE, 2021.

[146] Dingwen Zhang, Junwei Han, and Yu Zhang. Supervision by fusion: Towards unsupervised learning of deep salient object detector. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 4068–4076. IEEE Computer Society, 2017.

[147] Han Zhang, Tao Xu, and Hongsheng Li. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 5908–5916. IEEE Computer Society, 2017.

[148] Han Zhang, Ian J. Goodfellow, Dimitris N. Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 7354–7363. PMLR, 2019.

[149] Han Zhang, Zizhao Zhang, Augustus Odena, and Honglak Lee. Consistency regularization for generative adversarial networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

[150] Jing Zhang, Tong Zhang, Yuchao Dai, Mehrtash Harandi, and Richard I. Hartley. Deep unsupervised saliency detection: A multiple noisy labeling perspective. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 9029–9038. IEEE Computer Society, 2018.

[151] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016.

[152] 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.

[153] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient GAN training. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[154] Yang Zhao, Chunyuan Li, Ping Yu, Jianfeng Gao, and Changyou Chen. Feature quantization improves GAN training. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 11376–11386. PMLR, 2020.

[155] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2242–2251. IEEE Computer Society, 2017.

[156] 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, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pages 2814–2821. IEEE Computer Society, 2014.

[157] C. Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 391–405. Springer International Publishing, 2014. ISBN 978-3-319-10602-1.

[158] Wenbin Zou and Nikos Komodakis. HARF: hierarchy-associated rich features for salient object detection. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 406–414. IEEE Computer Society, 2015.