ABSTRACT

Title of Dissertation: Supervision and Data Dynamics in Vision Across Recognition and Generation Landscapes

Saksham Suri
Doctor of Philosophy, 2024

Dissertation Directed by: Professor Abhinav Shrivastava
Department of Computer Science
University of Maryland, College Park

This thesis looks at visual perception through the lens of supervision and data dynamics across recognition and generation landscapes. Generative and discriminative modeling form important pillars in computer vision. Depending on the task techniques to better learn and utilize the data and labels can change. Through this work we investigate different tasks along this landscape focusing on different supervision strategies, highlighting pitfalls in current approaches and propose modified architectures and losses to utilize the data better under different settings.

On the recognition side we start by analyzing Vision Transformers (ViTs) through a comprehensive analysis under varied supervision paradigms. We look at a mix of explicit supervision, contrastive self-supervision, and reconstructive self-supervision by delving into attention mechanisms and learned representations. We then look at a more specific case of supervision geared towards object detection which is called sparse supervision where their are missing annotations. We propose to utilize self and semi-supervised techniques to solve this task. Finally, we also

explore a discovery style framework with applications on GAN generated image detection. Unlike sparse supervision discussed earlier, this scenario handles the case where are test time we have an unknown number of new classes. We were the first work proposing this problem where instead of just identifying synthetic images, we also try to group them based on their generation source. The exploration of Generative Adversarial Networks (GANs) in an open-world scenario uncovers the intricacies of learning with limited supervision for discovery style problems.

On the generation side we delve into different supervision strategies involving decomposing and decoupling representations. In the first work we tackle the problem of paired Image-to-Image (I2I) translation by decomposing supervision into reconstruction and residuals and highlight issues with traditional training approaches. We then look at generating talking head videos through two different kinds of supervision, video and audio. For driving the generation using a video we look at decoupling representations for the task of few-shot talking-head synthesis where the supervision is provided using only a few samples (shots). For this task we factorize the representation into spatial and style components which helps the learning. To supervise the generation additionally through audio, we look at multimodal supervision for lip-synchronized talking head generation. For this we incorporate audio and video modalities to synthesize lifelike talking-heads which can work even in in-the-wild scenarios.

In the last part we showcase two works which link our experiences from generation and recognition where we explore generative modeling to improve recognition models. The first work here utilizes the advancements in diffusion based image generation models to improve recognition models. Given the high fidelity and control of generation which diffusion models have brought, we utilize synthetic data from these models and create a suitable pipeline to utilize this data effectively to improve detection and segmentation performance. As a follow up to our

ViT analysis we also propose a new technique to utilize off the shelf pretrained ViTs and generate high resolution features using a learnt lightweight feature transform. These high resolution features are especially effective for dense tasks like correspondence, segmentation, detection and object discovery.

Supervision and Data Dynamics in Vision Across Recognition and Generation Landscapes

by

Saksham Suri

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2024

Advisory Committee:
Professor Abhinav Shrivastava, Chair/Advisor
Professor Carol Y. Espy-Wilson, Dean's Representative
Professor Tianyi Zhou
Professor David Jacobs
Professor Andrew Zisserman, Oxford University

With profound gratitude, I thank my family and friends

for their unyielding and unconditional support.

Acknowledgments

This thesis has been possible with the guidance and collaboration of various people and teams, and I want to thank all who have contributed for their input.

First and foremost, I want to thank my advisor, Abhinav Shrivastava, who has been a pillar of immense support throughout my PhD. He has always been flexible with the topics and problems I work on, allowing me to pursue what I am most passionate about and interested in. I cherish all my discussions with him, which always left me in awe of how quickly he understands and suggests solutions to any research obstacles I encounter. Throughout my PhD, Abhinav motivated me to collaborate and eventually guide other students. I thank him for pushing me to do this, as it made my PhD more enriching. In addition to being a great mentor for research and academics, I have also learned a lot from him about how to live life and enjoy the PhD journey. He has been a constant listener for research and life problems and given valuable advice. There have been multiple moments throughout the PhD where he stepped up, going above and beyond to ensure my well being. I cannot thank him enough for his friendship, mentorship, and all the lessons I have learned from him throughout my journey.

I am also thankful to the multiple industry and academic mentors I have had, who have helped shape the way I think and tackle problems. Rama Chellappa, for admitting me to the PhD program at UMD and supporting me in my first year. Chenchen, Fanyi, Animesh, Sean, and Raghu at Meta, who guided me during my internship and provided a great experience. Ankan,

Vijay, Yuting, and Yusheng, who were great mentors during my Amazon internship. Mayank Vatsa and Richa Singh, who were my advisors during my undergrad and motivated me to pursue a PhD, sending me on this journey with a good training. My undergrad university, IIIT-Delhi and my school, Bluebells School International for being strong institutions providing holistic education and growth.

I want to thank DARPA SAIL-ON (W911NF2020009), DARPA SemaFor (HR001119S0085), DARPA GARD (HR00112020007), NSF and NIST Institute for Trustworthy AI in Law and Society (TRAILS) (2229885), and IARPA SMART (2021-21040700003) programs, Amazon Research Award to Abhinav, NSF CAREER Award (2238769) to Abhinav, independent gift from Facebook AI to Abhinav, Dean's Fellowship, Meta, and Amazon Research Labs, which helped me conduct my research.

I want to express my gratitude to my friends/collaborators (Sharath, Saketh, Matt W., Shishira, Matt G., Anubhav, Soumik, Vatsal, Shirley, Chuong, Daniel, Hanyu, Mara, Kamal, Max, Moustafa, Anshul, Ketul, Josh, Carlos) and friends/labmates (Pulkit, Archana, Namitha, Bo, Khoi, Nirat, Alex, Ahmed, Gaurav, Hao, Khoi, Lillian, Luyu, Navaneeth, Pallabi, Sonaal, Varuni, Vinoj, Yixuan, Shaan, Abhinav K., Vedant, Sai, Neha, Sneha, Samyadeep, Ahana, Roni, Ishita, Divya, Gowthami, Noor, Vasu, Shlok, Aman, Tushar, Dhruv) for their unwavering support and encouragement throughout this journey. Special shoutout to Saketh, Kamal, and Moustafa, who took me under their wings during my early years and were great mentors, helping me learn a lot about conducting good research. I also want to mention Sharath, who has been a great friend and collaborator, and has shown incredible resilience in the face of adversity, teaching me so much about living life.

I want to thank my parents, Dhiraj and Veena, for always supporting me throughout my life

and being pillars of strength, especially during my PhD. I thank both of you for raising me to be strong and teaching me keep pushing even in difficult times. All the education I have achieved today is because you provided me with the comfort and tools to pursue my goals. I want to thank my brother Saransh for being an amazing sibling and taking over so many responsibilities at home while I pursued my PhD. I am sure you will achieve great things in life. I also want to thank my fiancée, Marycait, who has been with me for the majority of my PhD. She stood by me through the ups and downs, providing unconditional support, patience, and understanding. Thank you for calming me down and being a pillar of emotional support. I would have not been able to complete this journey without you.

I also want to thank the administrative staff at UMD, including Tom Hurst, Migo Gui, Jodie Gray, Sharron McElroy, and multiple members of UMIACS and ISSS, who helped me through various steps and made my journey smoother.

My sincere apologies to those I've inadvertently missed. Without all of your support, guidance, and encouragement, this research would not have been possible.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Overview

Humans rely on a multitude of experiences, picking up bits of information from various encounters, using our instincts, and piecing together patterns to understand the world around us. Our ability to adapt to new scenarios, learn from a handful of examples, or infer missing details is a testament to the versatility of our learning mechanisms.

David Kolb once noted, "Learning is the process whereby knowledge is created through the transformation of experience." This sentiment encapsulates the essence of human learning, highlighting the transformative nature of experiences in shaping our understanding. Machines, however, face a different landscape. Teaching them to navigate the world's imperfections—data riddled with noise, missing elements, or just a few examples—requires specialized approaches. In the realm of computer vision, where machines seek to perceive and understand visual content, these challenges are particularly daunting. Consider the colossal volumes of visual data: mountains of images and videos that are not just tough to label due to their sheer quantity but even annotating them comes with their own set of issues. Some samples might have labels that aren't quite accurate, some might lack any annotations at all, while in other cases there might be very few samples for the rare categories, making it challenging for machines to learn effectively from them.

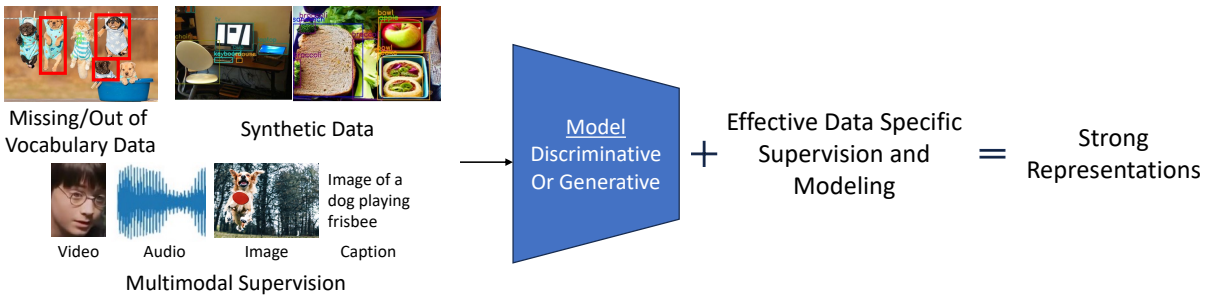


Figure 1.1: With different data setting that exist across recognition and generation tasks, understanding and using effective supervision and modeling strategies which account for the setup are important to get strong representations.

Self-supervised learning, semi-supervised learning, and few-shot learning — are some of the specialized paradigms designed to help machines deal with these very specific data challenges. Each of these approaches adapts how machines learn, mimicking in some ways how we, as humans, adapt to different situations by drawing on our past experiences or making intuitive leaps. And this is precisely what this thesis explores—the ways in which machines can better grapple with imperfect data and specialized learning methods across both recognition (where machines categorize and understand images/videos) and generation (where they create new visual content) domains within computer vision. Finally, by incorporating our expertise in both these paradigms, we couple them together to utilize generation to further improve recognition performance by either generating better features or generating synthetic data. By navigating these nuances, this work aims to understand and equip machines with better learning strategies, helping them make sense of the visual world more like us.

We divide and highlight the different topics this thesis covers in terms of the tasks and problems they deal with below.

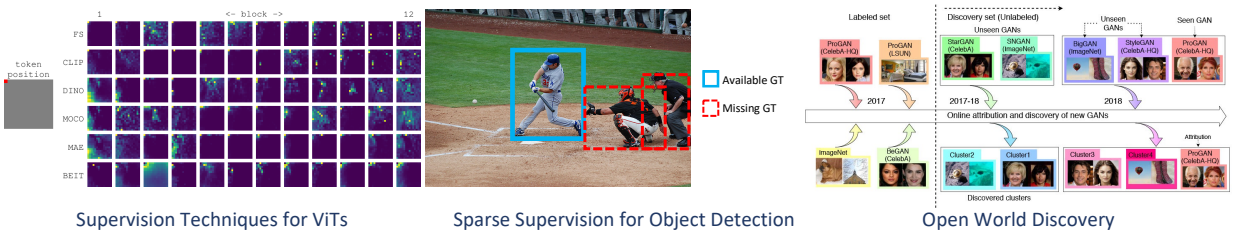


Figure 1.2: Different data and supervision strategies explored for recognition.

1.2 Recognition

We dive into recognition by first analyzing ViTs which have become the defacto backbone replacing the previously popular convolution neural networks (CNNs). We perform an extensive analysis of Vision Transformers (ViTs) operating under diverse supervision paradigms in Chapter 2. We look into explicit supervision, evaluating models under fully supervised setups alongside innovative approaches like CLIP. Additionally, we look at contrastive self-supervision models such as DINO and MoCo, alongside reconstructive self-supervision methods like BEiT and MAE. This comprehensive study aims to unveil how distinct supervision techniques shape ViTs' behavior, elucidating the adaptability of these models across various learning methodologies. Expanding our scope, we address sparse supervision, a facet crucial in object detection scenarios in Chapter 3. Annotations might be missing or incomplete due to occlusion, out of vocabulary instance or missed annotation due to repetitions. Our proposed strategy blends self and semi-supervised techniques, accompanied by architectural adjustments, safeguarding detector training from penalizing absent annotations and fortifying the model's robustness against sparse data.

Continuing this pursuit, this thesis introduces a pioneering discovery-style framework tailored to identify the source or class of synthetically generated images—an innovative step in

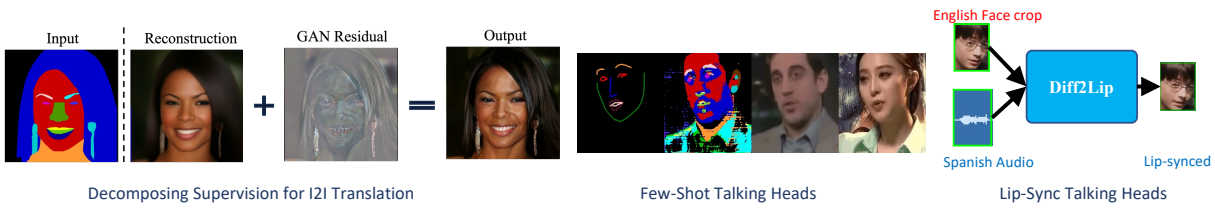


Figure 1.3: Different data and supervision strategies explored for generation.

open-world scenarios in Chapter 4. Here, test images originate from an unknown array of Generative Adversarial Networks (GANs) without prior class information. An iterative algorithm refines features through clustering, effectively improving the features over time. This enables the identification of images from previously unknown GAN sources, despite the absence of explicit class information, offering a unified approach within the spectrum of supervision challenges in visual recognition. Figure 1.2 highlights the main works we look at in this part.

1.3 Generation

Exploring the landscape of generation, our first work in Chapter 5 addresses the challenges within paired Image-to-Image Translation, where we identify a major issue residing in the conflicting nature of current losses applied to this task. By decomposing supervision into reconstruction and residuals, we aim to resolve these conflicts, offering a new yet simple insight that enhances the realism of output images. This critical change highlights the limitations of traditional training methods and presents a promising avenue for more realistic image generation by letting the model learn the multi-modal nature of outputs.

Moving forward we explore two works along the lines of talking head synthesis in Chapter 6. The first looks at few-shot talking-head synthesis, a task where limited supervision is provided through just a few samples in a few-shot setting. Here, we introduce a new method that

Figure 1.4: Utilizing generation to improve recognition by generating better features and more data.

disentangles representations into spatial and style components. This approach significantly improves results both qualitatively and quantitatively, demonstrating the usefulness of decoupling representations while navigating few-shot learning scenarios. This work not only refines the synthesis process but also sets the stage for more effective and adaptable learning from minimal data. In the next work we also condition our talking heads on audio to maintain lip-synchronization making it a case of multimodal supervision. Integrating both audio and video modalities, our method synthesizes remarkably realistic talking-heads. The synthesis achieved here maintains good quality, enabling seamless alterations in audio or subjects while preserving the integrity and coherence of generated content. This innovative fusion of audio and visual cues establishes using diffusion models shows the power of these models for such challenging tasks, reshaping the landscape of lifelike content generation. Figure 1.3 highlights the main works we look at in this part.

1.4 Generation for Recognition

Finally, in our venture through multiple tasks across generation and recognition, we explore how we can use ideas from generation to improve recognition models. This prompts an intriguing query—can the advanced capabilities of diffusion models be harnessed to enhance recognition tasks? This question drives our exploration into utilizing grounded diffusion models to generate object-centric data in Chapter 7. Through our work we show that synthetic data from diffusion

models can help improve performance for object detection and segmentation models. Such data is especially effective in long tailed and low data regimes. Our proposed pipeline offers a modular approach to generate, filter and utilize synthetic data effectively.

Moreover, we also look back at our study of Vision Transformers (ViTs), particularly the constraints posed by low-resolution spatial features generated by ViTs. To counter this challenge, we propose a simple and lightweight technique to utilize self-supervised learning to enrich the spatial feature resolution of ViTs without requiring training of the ViT model itself in Chapter 8. Our proposed methodology involves employing a simple lightweight network to refine and amplify the spatial resolution of ViT features by learning to generate higher resolution features conditioned on the low resolution features and image. This approach shows to be effective in improving performance of tasks using these features while incurring minimal training and inference computation costs. Figure 1.4 highlights the main works we look at in this part.

To summarize, we start by analyzing different supervision strategies for ViTs in Chapter 2. This is followed by more specific supervision scenarios for sparse annotation in Chapter 3 and open world learning in Chapter 4. We then look at different supervision strategies in generative modeling starting with the task of paired image-to-image translation in Chapter 5. We also look at two different supervision strategies for talking head synthesis in Chapter 6 where we generate talking head videos conditioned on videos as well as audio. Finally we present two works to utilize our experience and intuitions from generative modeling to improve recognition. Chapter 7 looks at providing supervision using synthetic data to improve object detection and segmentation performance. While Chapter 8 dives into generating high resolution ViT features in an efficient and cheap manner. At the end in Chapter 9, we discuss some future directions and conclude the thesis.

Part I: Data and Supervision Strategies in Recognition

Chapter 2: Teaching Matters: Investigating the Role of Supervision in Vision Transformers

The field of Computer Vision has advanced massively in the past decade, largely built on the backbone of Convolutional Neural Networks (CNNs). More recently, Vision Transformers (ViTs) [15] have shown the potential to overtake CNNs as the go-to visual processing model. Prior works have asked the question *do ViTs see like CNNs do?* [16], but in this work, we ask: *how do ViTs learn under different supervision?* Past examinations of ViTs have largely focused on models trained through full supervision. Instead, we aim to characterize the differences and similarities of ViTs trained through varying training methods, including self-supervised methods. Unlike CNNs, the ViT architecture imposes few structural biases to guide the learning of representations. This gives them the flexibility to learn diverse information processing strategies, and through our analyses, we uncover a wide array of ViT behaviors.

There are countless ways to analyze ViTs, so to guide this analysis we choose three major domains which correspond to the *How*, *What*, and *Why* of ViTs. For the *How*, we focus on *how* ViTs process information through **Attention**. Multi-Headed Attention (MHA) layers are arguably the key element of ViTs, and they most distinguish them from CNNs. For the *What*, we examine the **Features** of ViTs, as these are typically *what* practitioners take away from them. Finally for the *Why*, we focus on **Downstream Tasks**, which are *why* we care about using ViTs.

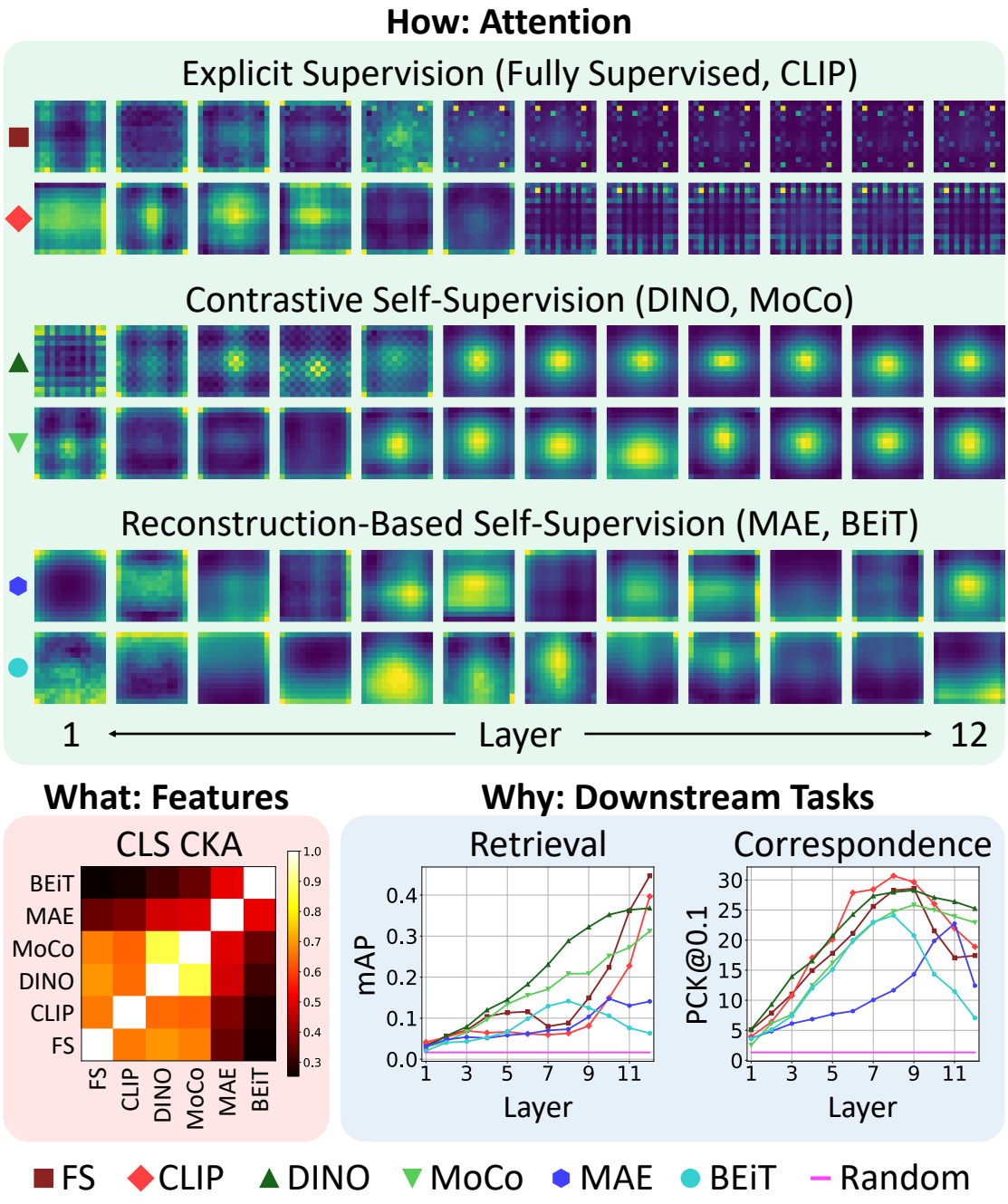


Figure 2.1: **ViTs exhibit highly varied behaviors depending on their method of training.** In this work, we compare ViTs through three domains of analysis representing the How, What, and Why of ViTs. **How** do ViTs process information through attention? (Top) Attention maps averaged over 5000 images show clear differences in the mid-to-late layers. **What** do ViTs learn to represent? (Left) Contrastive self-supervised ViTs have a greater feature similarity to explicitly supervised ViTs, but also have some similarity with ViTs trained through masked reconstruction. **Why** do we care about using ViTs? (Right) We evaluate ViTs on a variety of global and local tasks and show that the best model and layer vary greatly.

Our work unveils that a powerful aspect of the ViT architecture is its local-global dual nature, which plays a role in all three aspects of our analyses. While standard CNNs are restricted to building representations hierarchically from local to global, in a ViT each token can attend to information from any other image region at any time. And unlike popular CNN modifications like Spatial Pyramids [17, 18, 19, 20] and top-down strategies [21, 22, 23], ViTs have the freedom to decide when and where global information should be integrated. In this study, we show that the order and the relative ratio of local and global attention in ViTs varies dramatically based on the method of supervision. We also find clearly different trends in the allocation of attention in the mid-to-late layers of these networks, as highlighted in Figure 2.2. This local-global dual nature is also embedded into the structure and features of the ViT, which encodes both local spatial tokens and a non-local classifier (CLS) token throughout its entire depth. We analyze the features of ViTs for both the CLS and spatial tokens, and assess how they align with semantics at the image, object, part, and pixel-level. We perform this analysis at every layer of the ViT to show the emergence of different levels of semantic information. Finally, we assess ViTs on a number of local and global downstream tasks.

Overall, our contributions are: **[1]** A detailed comparison of ViTs trained with six different methods, including both fully supervised and self-supervised training. **[2]** A cross-cutting analysis spanning three major domains: Attention, Features, and Downstream Tasks. **[3]** Multiple insights into the inner workings of ViTs to guide future development of ViT variants, training strategies, and applications.

In addition, we summarize some of our key observations about ViT behavior: **[1]** The attention maps of explicitly supervised ViTs devolve into **Sparse Repeating Patterns** in the mid-to-late layers, but the quality of features continues to improve in these layers (Section 2.3.1). **[2]** All

ViTs studied learn to use **Offset Local Attention Heads**, suggesting they are fundamentally necessary in ViTs (Section 2.3.2). To the best of our knowledge, no prior work has brought attention to this phenomenon. **[3]** ViTs learn to process local and global information in different orders depending on their method of supervision (Section 2.3.3). **[4]** All ViTs studied differentiate salient foreground objects by the early-to-mid layers (Section 2.3.4). **[5]** Reconstruction-based self-supervised methods can learn semantically meaningful CLS representations, even when the CLS token is only a placeholder (Section 2.4.1, 2.4.2). **[6]** Supervised method's features are the most semantically rich, but contrastive self-supervised methods are comparable or even superior in some cases (Section 2.4.2, 2.4.3). **[7]** For localized tasks, the best performance often comes from a mid-to-late layer (Section 2.5.2). **[8]** There is no single "best" training method or layer for all downstream tasks (Section 2.5.3).

2.1 Related Work

Previous works have attempted to understand the representation quality for both supervised and self-supervised training for Convolutional Neural Networks (CNNs). [24] focuses on understanding the concepts learned by individual neurons while [25] looks at explaining their compositionality in the case of supervised networks. Simultaneously, due to the popularity of self-supervised learning methods [26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37] multiple works have analyzed these representations learned from no labels. Under this umbrella, [38, 39] tried to understand the effect of training data in terms of both the number and type of samples. Some works [40, 41] analyze the alignment, separability, and uniformity of features while [42] looks at invariance to augmentations like occlusion, illumination, and viewpoint change in the learned

representation. [43] looks at the downstream performance of self-supervised networks on fine-grained tasks. Finally, [44, 45, 46] analyze multiple self-supervised methods and compare their performance based on representation similarity and downstream task performance over multiple datasets along with comparisons to supervised methods.

Since the proliferation of ViTs, a number of works have tried to understand and explore the different properties of the representations learned by these networks. A few works [47, 48, 49] have analyzed the robustness of ViT features against corruptions, perturbations, distribution shifts, and adversarial examples while also analyzing the role of self-attention for robustness. [50] benchmarks different pretrained ViTs as backbones for object detection. [51] provides a theoretical understanding of how MAEs work while [52] analyzes attention using convex duality. [53] gives insights to train and use ViTs more efficiently. [54] gives a deeper understanding of how Multi-Headed Attention layers work while comparing and contrasting to how convolution layers behave in terms of the loss landscapes and low-pass/high-pass filtering. [16] compares fully supervised ViTs and ResNets in terms of the local and global information encoded at different depths, the role of skip connections, and the uniformity of representations.

All these prior works either examine the impact of supervision on CNNs or compare CNNs and ViTs trained with full supervision. Some recent and concurrent works have compared the properties of differently supervised ViTs, though typically focused on a particular task and only two methods of supervision at a time. [55] compares the properties of fully supervised and DINO ViT features in the context of dense feature descriptors, and [56] further compares these two across several semantic correspondence tasks. [57] compares fully supervised and CLIP ViTs through feature visualizations. To the best of our knowledge, we present what is to date the broadest and the most in-depth comparison of ViTs with varying supervision, including six different

methods covering three supervision subcategories. Additionally, we propose new attention-based analysis methods along with evaluations on multiple downstream tasks focused on both local and global information.

2.2 Experimental Design

2.2.1 A Primer on Vision Transformers

Vision Transformers (ViTs) [15] are adapted from Transformers [58] for the Natural Language Processing domain. A ViT consists of an array of tokens, each representing an image patch. In addition, most ViTs include an extra "classifier" or "CLS" token, which is connected to the task-specific output layers during training. ViTs use Multi-Headed Attention (MHA) layers [58], which use a Query-Key-Value system that allows each token to attend to all other tokens with a variable intensity attention map. This is in stark contrast to the limited receptive fields of convolutions. These layers are "multi-headed" because they repeat this process multiple times in parallel, allowing tokens to apply multiple attention strategies concurrently. A ViT architecture includes multiple blocks, each with one MHA layer followed by a position-wise fully connected layer. Unlike CNNs, which usually get narrower in deeper layers, ViTs maintain the same "width" (number of tokens) throughout. There are some transformer variants, like SWiN transformers [59], that introduce a narrowing width, but for our analyses, we focus on only traditional ViTs. Specifically, our primary analysis focuses on ViT-Base models with patch size 16×16 (ViT-B/16) and input size 224×224, which results in a 14×14 spatial token array. ViT-Base has 12 blocks and 12 attention heads per MHA layer.

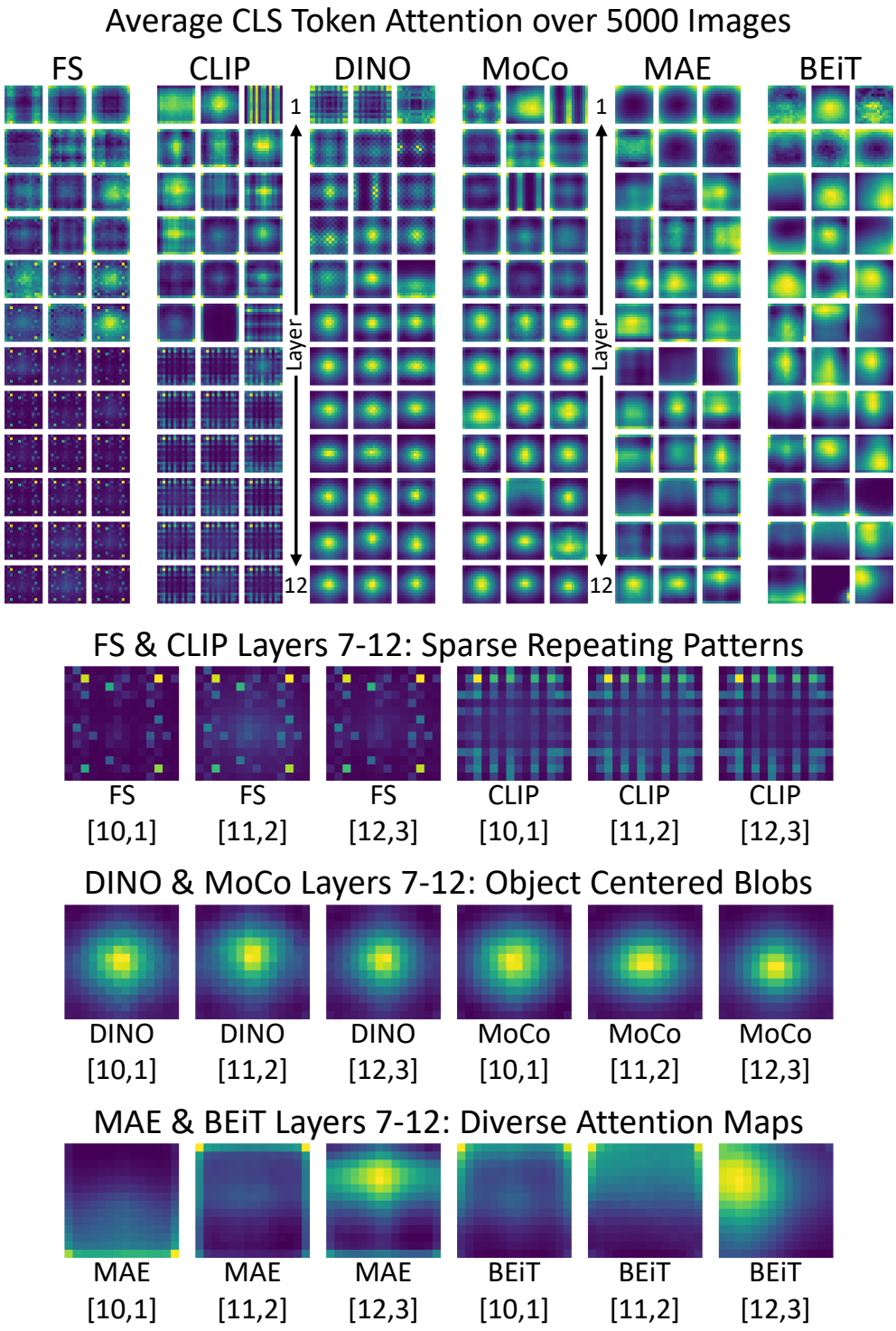


Figure 2.2: **Clear differences in attention emerge in the mid-to-late layers under different supervision methods**. These plots show the attention maps of CLS tokens averaged over 5000 ImageNet images. Rows indicate layers and columns indicate heads. For brevity, we show only three heads per layer. The bracketed numbers in the lower half denote the layer and head.

2.2.2 Methods of Supervision

Although a large number of ViT training methods have been proposed in a short span of three years, many of the most popular methods can be loosely categorized into the following three groups. From each group, we select two representative models for in-depth analysis.

Explicit Supervision. These models are trained with an explicit objective that is defined either by human annotations or by labels derived from another source, like paired image captions. For this category, we use a Fully Supervised (FS) ViT pretrained on ImageNet21k and fine-tuned on ImageNet1k [60, 61], as well as a CLIP ViT [62].

Self-Supervision (Contrastive). Self-supervised learning methods broadly attempt to train a model through a pretext task that can be directly derived from the input data. Among the more popular pretext tasks are contrastive learning methods [27, 63] which generally present a model with multiple augmented views of the same image alongside distractor views of other images. The model must learn to identify which of the views came from the same image. For this category, we select DINO [26] and MoCo-v3 [64] which we denote simply as MoCo for the rest of this paper.

Self-Supervision (Reconstruction). Another popular category of self-supervision is reconstruction methods, which train models to predict the missing content from masked or otherwise corrupted images. We select MAE [28] and BEiT [29] for this category. Note that MAE has a separate decoder which is discarded after pretraining, while BEiT's decoder is learned in the same ViT. This has a strong impact on the behavior of the later layers of BEiT.

2.2.3 Datasets

We study the ViTs on multiple datasets and downstream tasks. Unless otherwise specified, we use ImageNet-50 [65], a subset of ImageNet [66] which narrows the dataset down to 50 representative categories. We sample 100 images per class to create a diverse collection of 5000 images. We additionally use PartImageNet [67] to measure Attention Saliency and part-level feature purity, as well as COCO [68] to measure object-level feature purity. We use revisited [69] Oxford [70] (ROxford5k) for evaluating image retrieval, DAVIS [71] for video segmentation, and SPair-71k [72] for keypoint correspondence.

2.2.4 Proposed Analyses

Our analysis is broadly divided into three domains covering the *How*, *What*, and *Why* of ViTs:

***How* ViTs process local/global information (Attention).** Do self-attention heads learn to operate in different ways depending on their method of training? Are there distinctive modes of attention behavior? How does supervision impact the processing order of local and global information?

***What* we take away from ViTs (Features).** How do the final and intermediate representations of a ViT change depending on the method of supervision? Are these trends similar or different for CLS *vs*. spatial tokens?

***Why* we use ViTs (Downstream Tasks).** Which forms of supervision are best suited for different downstream tasks? Which layers of a ViT produce features that are best for different local and global tasks?

2.3 Attention Analysis

Multi-Headed Attention layers are one of the defining components of the Transformer architecture, and the attention maps they generate can give key insights into what is similar or different about ViTs trained through different methods. We perform an in-depth examination of the self-attention maps of ViT-B/16 models at every layer. Through this study, we uncover a diverse range of attention head behavioral modalities.

2.3.1 Attention Visualizations

We start by examining the attention maps of the CLS tokens of each head and layer. To gain a comprehensive understanding of each head's behavior, we compute the average attention maps over 5000 ImageNet images, as shown in Figure 2.2. For brevity, we display only three heads per layer.

One of the clearest differences can be seen by comparing the mid-to-final layers. For the contrastive self-supervised methods, DINO and MoCo, the attention maps tend to be centered blobs. These heads tend to focus on salient foreground objects, so these blobs simply reflect object-centered photography bias. For the reconstruction-based methods, MAE and BEiT, we see a more diverse group of attention maps. This is likely because these methods must reconstruct all image regions, and thus their attention in the final layers must be more diverse and cover more of the image. Finally, for the explicitly supervised methods, FS and CLIP, the mid-to-final layers do not focus on salient object regions and instead focus on **Sparse Repeating Patterns** with seemingly no spatial meaning. This occurs for both the CLS tokens and spatial tokens, and the patterns are repeated across both heads and layers. We hypothesize that these patterns occur

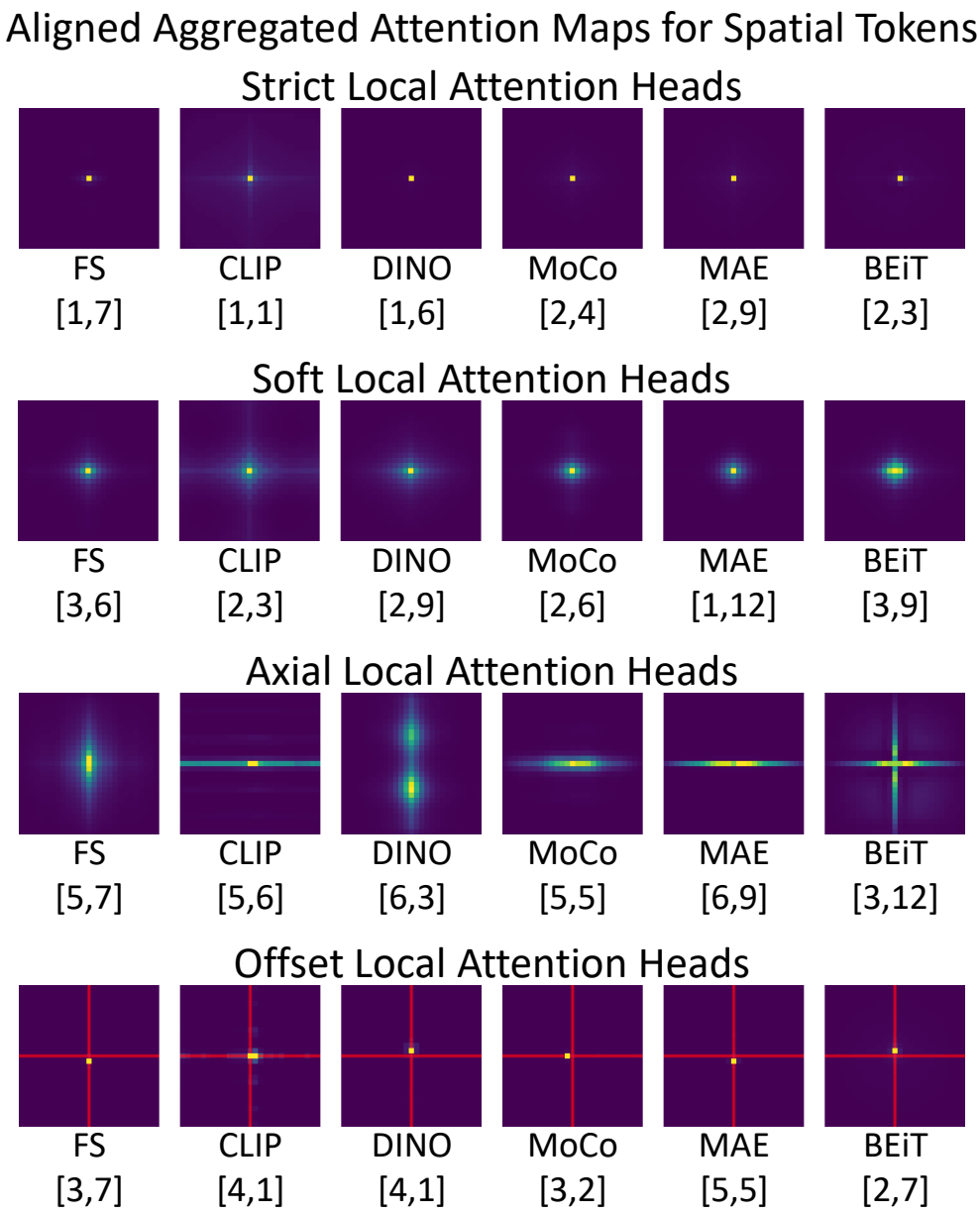


Figure 2.3: **Multiple distinct forms of local attention exist.** We visualize spatial token attention using Aligned Aggregated Attention Maps, and highlight different types of local attention heads, including Strict, Soft, Axial, and Offset Local Attention Heads. In row 4 we draw the mid-lines in red as a visual aid.

because the mid-to-late layers are no longer focused on parsing the scene structure, and instead are using their processing power to generate their final decisions for their respective tasks. This phenomenon helps to explain why the attention maps of the later layers of fully-supervised ViTs are poorly suited for segmentation tasks, as was observed by [26].

2.3.2 Emergence of Offset Local Attention Heads

It has been shown that ViTs use a mixture of short-range "local" and long-range "global" attention heads in any given layer [15, 26]. To gain a better understanding of local attention, we propose a visualization strategy of **Aligned Aggregated Attention Maps (AAAMs)**. We extract all spatial token attention maps for 5000 ImageNet images, but before averaging them, we first realigned them so the current spatial token is always in the center of the array. Studying these aligned views reveals multiple forms of local attention, shown in Figure 2.3. We find Strict Local Attention Heads, which attend almost completely to their own position, as well as Soft Local Attention Heads, which attend to a wider neighborhood around them. We also find Axial Local Attention Heads, which are elongated to attend to the local neighborhood along one or both spatial axes.

But perhaps the most noteworthy type of head we observe is the **Offset Local Attention Head**. These are heads that attend locally, but to a point or region offset from the current token in a vertical or horizontal direction. We find instances of Offset Local Attention Heads in all the models examined, suggesting they are fundamentally necessary for ViTs. To the best of our knowledge, ours is the first work to draw attention to this phenomenon. We believe that such heads are absolutely necessary because ViTs, unlike CNNs, do not have an easy built-in

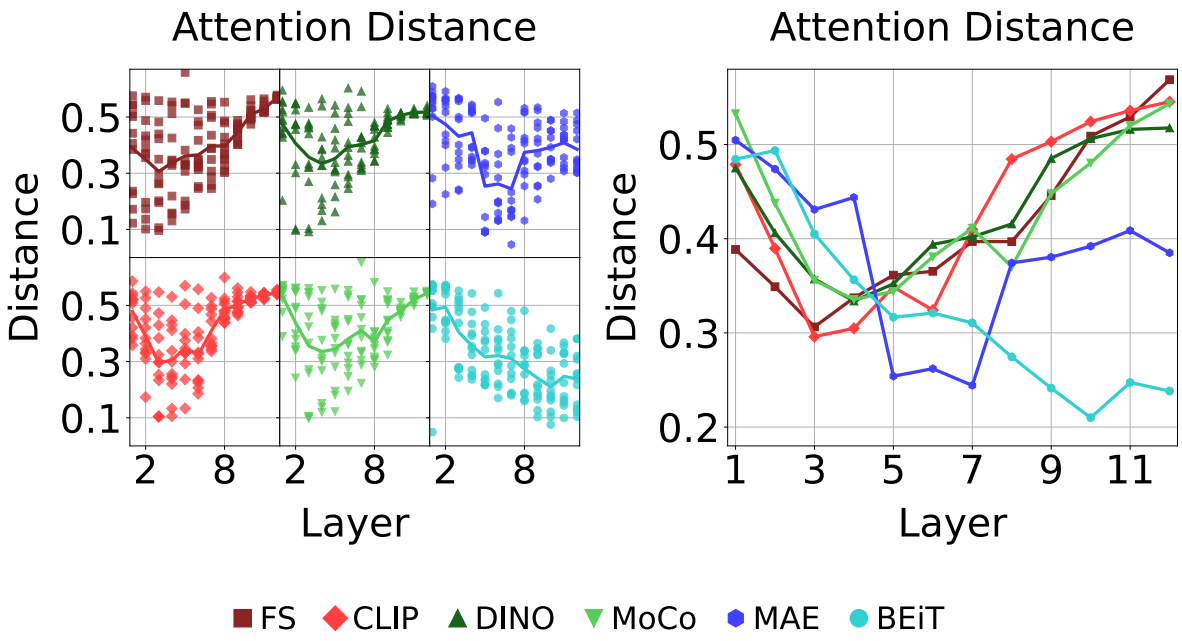


Figure 2.4: **Different methods of supervision lead to different orderings and ratios of local and global processing.** We show the Average Attention Distance of all ViT attention heads organized by layer (left), and the per-layer averages (right).

way to test if two features occur next to each other *with a particular spatial arrangement*. In a CNN, this type of check is naturally embedded into the convolution operator. But in a ViT, there is no inductive bias to induce such a check. For comparison, Soft Local Attention Heads are able to identify if a certain feature is near another feature, but they cannot identify their specific directional arrangement due to their symmetrical attention pattern. The existence of Offset Local Attention Heads implies one possible path for improvement for the ViT architecture, possibly by adding a self-attention variant that introduces some implicit directional structure.

2.3.3 Average Attention Distance

We measure the Average Attention Distance [15, 16] of each head to assess if particular heads have a short-range "local" focus or a long-range "global" focus. This metric is computed by measuring the distance from each spatial token to all other tokens and taking a weighted

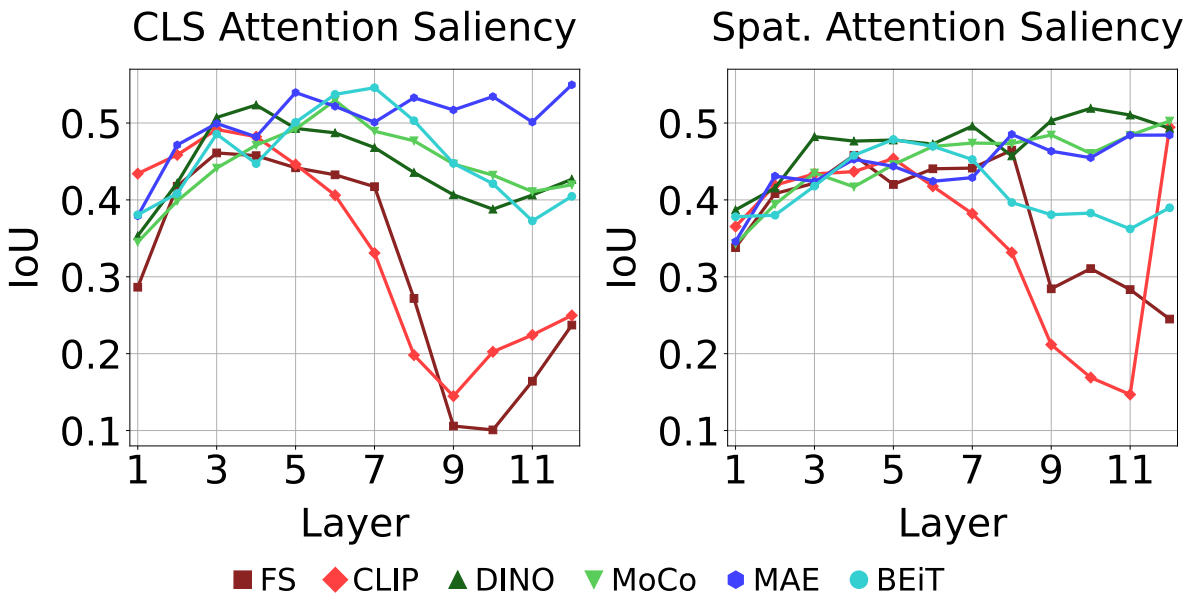


Figure 2.5: **Attention IoU with salient content plateaus early for all ViTs evaluated.** We calculate the alignment of ground-truth segmentation masks with CLS token attention maps (left) and the average of spatial token attention maps (right).

average using the attention map. We normalize the distances so the token grid is embedded on a 1×1 square. [16] observed that for a well-trained fully-supervised ViT, the early layers have a mixture of local and global attention heads, while the later layers have only global attention heads.

Figure 2.4 (left) shows the Average Attention Distances of all heads organized by layer and model. Like [16], we see that most layers use a mixture of local and global attention heads, however, we also find that the ordering of local and global processing varies greatly with the supervision type. FS, CLIP, DINO, and MoCo all use exclusively global attention heads in the last layers, but the reconstruction-based methods MAE and BEiT use a diverse range of heads in their later layers. Figure 2.4 (right) compares the combined Average Attention Distances at the per-layer level. In all models, we observe a greater number of global attention heads in the initial layers, followed by decreased distances around layers 3-6. This result is again in contrast

to [16]. The behaviors diverge in the mid-to-late layers. For the models trained with explicit or contrastive supervision, the Attention Distance trends upward in the later layers. For the reconstruction-based methods, the Average Attention Distances stay lower. These results show that, unlike CNNs, ViTs can learn a variable local/global processing order depending on the training method used.

2.3.4 Attention Alignment with Salient Content

One of the most desirable (and exploitable) features of DINO is that the CLS token attention maps of the last layer tend to be well-aligned with salient foreground objects [26]. Several methods propose to use DINO attention maps, feature maps, or a combination of the two to generate segmentations in a self-supervised manner [73, 74, 75]. We conduct a quantitative analysis of this property at all layers of the ViTs, both to measure the usefulness of masks and to assess how early the ViTs differentiate salient object regions. Like [26], we threshold the CLS token attention masks keeping 60% of the total attention mass. We then compute the Intersection over Union (IoU) of said masks with ground-truth segmentations from PartImageNet [67]. As an alternative to CLS token attention, we also extract masks using the average of spatial token attention maps. We present results for the single "best" head per layer in Figure 2.5.

We see a clear drop in FS and CLIP mask IoU around the middle of the network, which directly corresponds to the emergence of the Sparse Repeating Patterns observed in Section 2.3.1. We also find that the IoUs plateau around layers 3-6 for all networks. This demonstrates that ViT models already have a solid understanding of foreground/background separation by the middle layers. While the later attention maps of FS and CLIP are much worse than their self-supervised

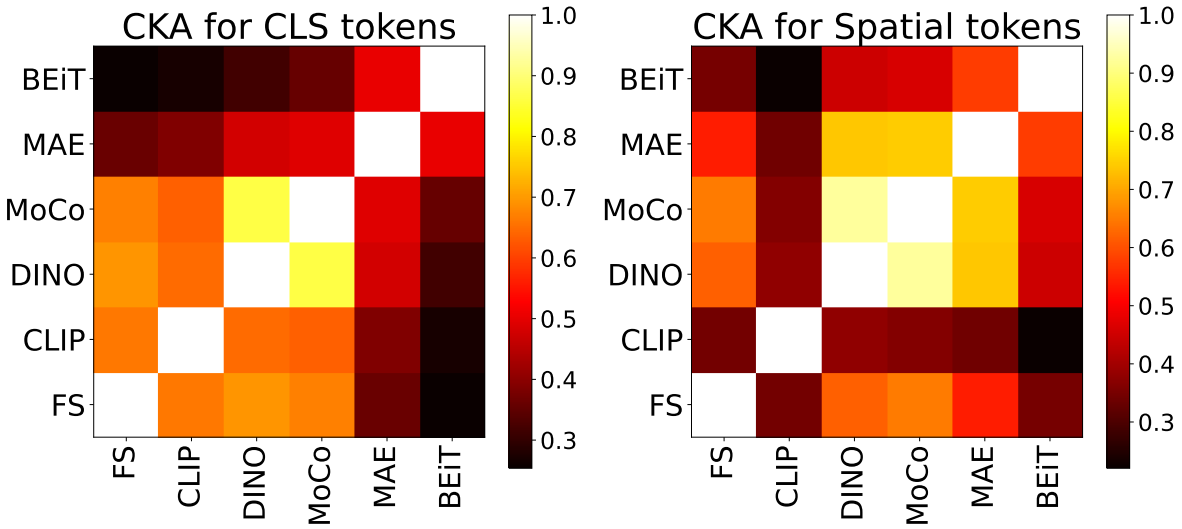


Figure 2.6: CKA similarity between final layer features of different ViTs for their CLS tokens (left) and spatial tokens (right).

counterparts, their early-to-mid layers are more comparable. We find that MoCo, MAE, and BEiT can all produce attention maps with IoUs that are comparable with DINO. In addition, we see that the average of spatial tokens produces maps that are comparable with the CLS token, and for CLIP the IoU increases greatly in the final layer.

2.4 Feature Analysis

In this section, we directly compare ViT features across models and layers using Centered Kernel Alignment (CKA) [76, 77]. We also study unsupervised clustering performance to compare global and local semantic information in the learned representations.

2.4.1 Last Block Feature Comparisons

Comparing representations is non-trivial due to varying feature sizes, large feature representations, and lack of alignment between them. To overcome this, we use batched Centered

Kernel Alignment (CKA) [76, 77, 78] which can align features and compute a similarity score. We compare the last layer outputs for each model.

Figure 2.6 (left) shows that the CLS token representations are usually similar for similar supervision strategies (explicit, contrastive, reconstruction). The contrastive methods, MoCo and DINO, show very high similarity to each other, indicating that the CLS token encodes the same type of information for both these methods. There is also an increased level of similarity between the explicitly supervised methods, FS and CLIP, and the contrastive methods. Interestingly, we see that MAE has as high a similarity with DINO and MoCo as it does with BEiT. This result is surprising because MAE's CLS token has no explicit training objective or loss, and the way these approaches are trained is very different. This presents evidence that training autoencoders with a high masking percentage indeed forces the model to learn image-level semantics.

In Figure 2.6 (right), we look at the similarity of the last-layer spatial token representations. Unlike the CLS token representations, CLIP and FS have low similarity in their spatial representations. The self-supervised methods DINO, MAE, and MoCo show a high level of similarity to each other, and a lower level of similarity to BEiT. MoCo and DINO show the highest similarity due to their similar kind of self-supervision. Once again, MAE has a high similarity to MoCo and DINO despite their very different supervision.

2.4.2 Feature Clustering for Global Semantics

Through this analysis, we aim to test how well the learned CLS and spatial token representations encode global (image-level) semantic information at every layer. We extract the CLS token features from the end of each block for 5000 ImageNet images, and we generate k-Means

cluster assignments with $k = 50$. We present results for cluster purity measured with respect to ground truth image labels. For the spatial tokens, we follow the same process except we average-pool over all positions before clustering. We also compute a random chance score by replacing the ViT features with Gaussian random noise.

For CLS token features, shown in Figure 2.7 (left), cluster purity improves with depth with the exception of the last layers of BEiT. This is likely because the last layers of BEiT serve as a task-specific decoder, unlike MAE, where the decoder is separate and discarded after pretraining. Unsurprisingly, FS achieves the best cluster purity, followed by CLIP. The contrastive methods, DINO and MoCo, achieve scores close to the explicitly supervised methods. The reconstruction-based methods, MAE and BEiT, have the lowest cluster purity, but they are still above random chance, which again indicates that they do learn to encode some image-level semantic information in their CLS tokens. Also, we find that semantic information emerges earlier for DINO and MoCo. For the spatial token features, shown in Figure 2.7 (right), the cluster purity of FS rises earlier compared with the FS CLS token. This suggests that the FS spatial tokens do more work gathering semantic information in the early layers. For all other ViTs, the spatial feature purity is lower in the final layers, but is comparable in layers 1-7.

2.4.3 Feature Clustering for Local Semantics

We next measure how well the spatial token features differentiate salient image content at the object or part-level using COCO [68] and PartImageNet [67] respectively. We use a tiling strategy to extract a denser array of features. Using ground truth segmentation masks, we extract and average the features of the tokens overlapping with the masks. This generates a collection

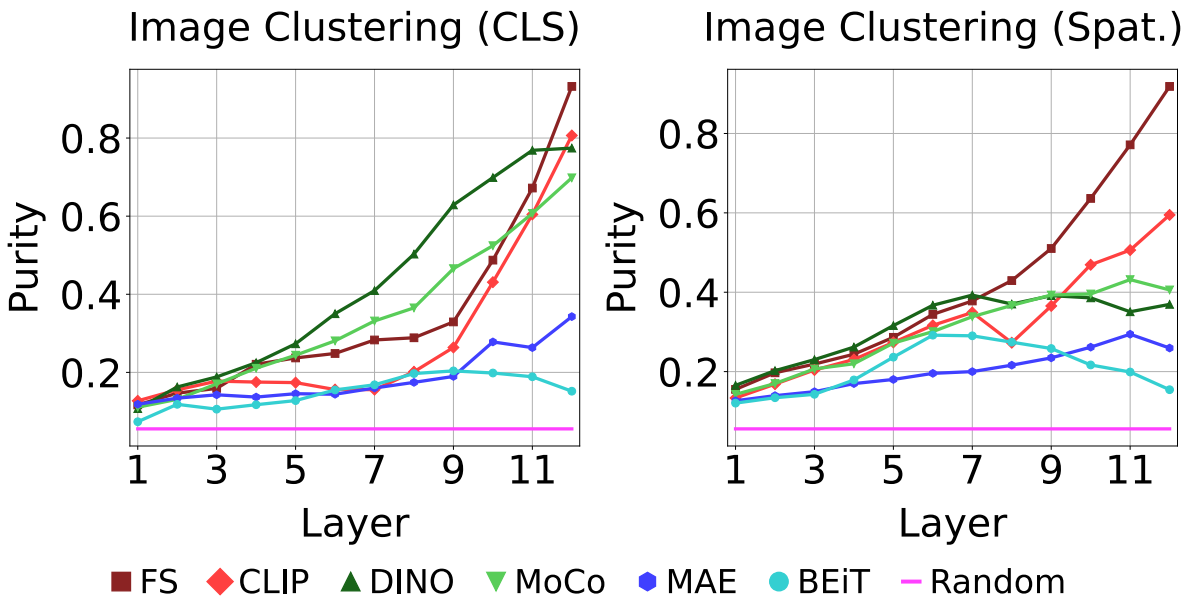


Figure 2.7: Clustering purity analysis with image-level labels in ImageNet-50 for CLS features (left) and average-pooled spatial token features (right).

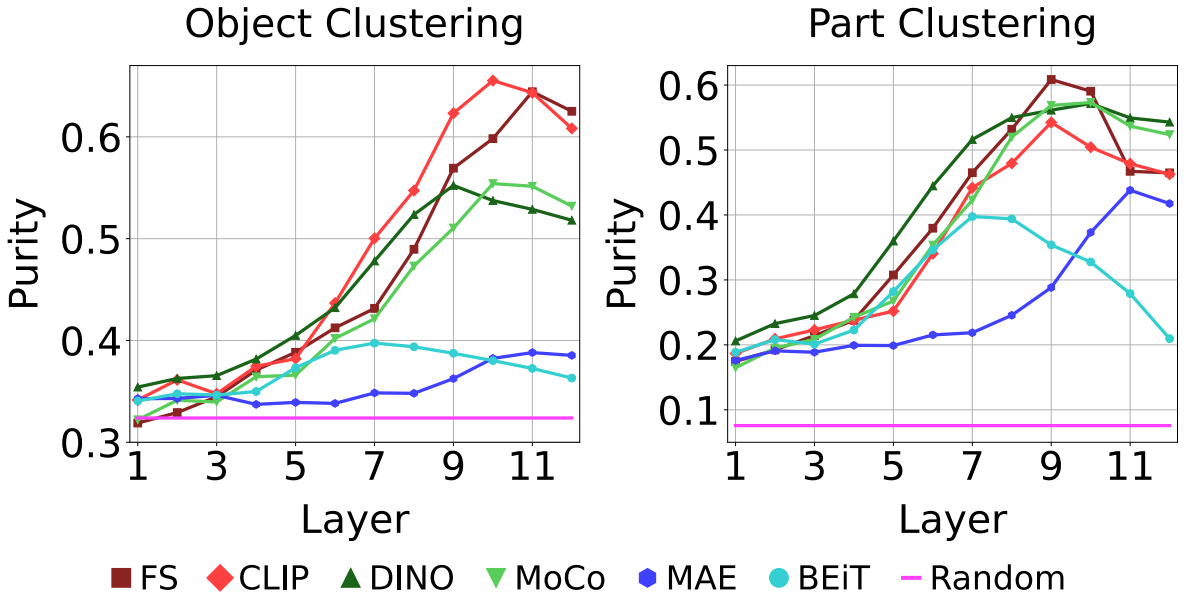


Figure 2.8: Clustering purity of spatial token features gathered at the object-level in COCO (left) and part-level in PartImageNet (right).

of object-level or part-level features which we cluster just like Section 2.4.2. The results for object-level features are shown in Figure 2.8 (left). We see that the supervised methods CLIP and FS have the highest feature purity by far, followed by the contrastive methods DINO and MoCo. The purity is much lower for the reconstruction methods MAE and BEiT. For part-level features, shown in Figure 2.8 (right), FS achieves the best purity, but the contrastive methods are very competitive in this case, surpassing CLIP completely. In addition, while still being the lowest scoring, MAE and BEiT are much more competitive at the part-level. Like the image-level purity, the object and part-level feature purity tend to improve with depth, but the purity peaks early around layers 9 to 11. The peak for BEiT is even earlier, likely due to its integrated decoder.

2.5 Downstream Task Analysis

Finally, we analyze the performance of these models on downstream tasks that can be performed directly without any fine-tuning or training. We follow the evaluation protocols of [26, 79] for k-NN classification, image retrieval, and video object segmentation. We also perform keypoint correspondence as a more local-focused task. Again we compute random chance scores by replacing all ViT features with Gaussian noise.

2.5.1 Global Tasks

ImageNet Classification. We perform k-Nearest Neighbor (k-NN) image classification on ImageNet [66] with $k = 20$. We use the CLS token features from each network and assign the label for a test sample based on the training set features and labels. As can be seen from

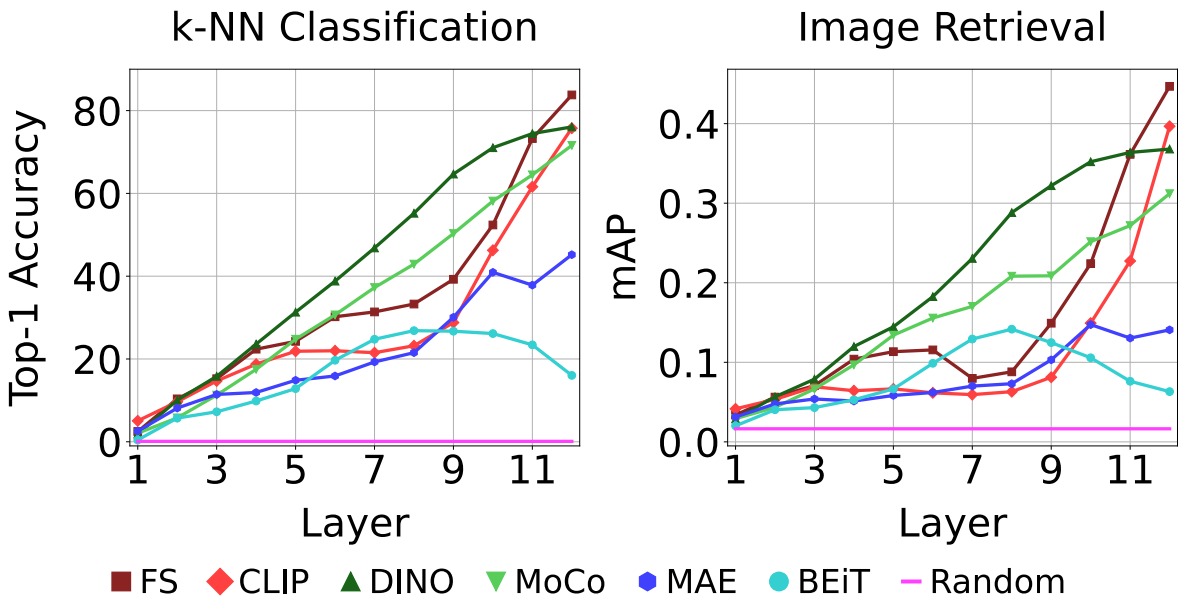


Figure 2.9: Global (image-level) downstream task analysis using the CLS token. We present k-NN classifier Top-1 Accuracy on ImageNet (left) and image retrieval mAP on ROxford5k (right).

Figure 2.9 (left), FS performs the best as it has been trained to classify the same dataset. DINO and MoCo follow a similar trend as Section 2.4.2 and better encode semantic information in the earlier layers. FS and CLIP also follow a similar trend where their performance shoots up in the last few layers. It is also interesting to see how MAE and BEiT, for which the CLS tokens have no explicit objective, do better than chance, although MAE is considerably better than BEiT.

Image Retrieval. Similar to k-NN classification, we utilize the CLS token representation for retrieval. We evaluate on ROxford5k [69] for the Medium (M) split and report the Mean Average Precision (mAP). The results, shown in Figure 2.9 (right), align closely with those for k-NN Classification on ImageNet. FS performs the best followed by CLIP and then DINO and MoCo, and finally by MAE and BEiT with the lowest performance. We hypothesize that the local/global crops used in DINO training help it perform competitively in these global tasks.

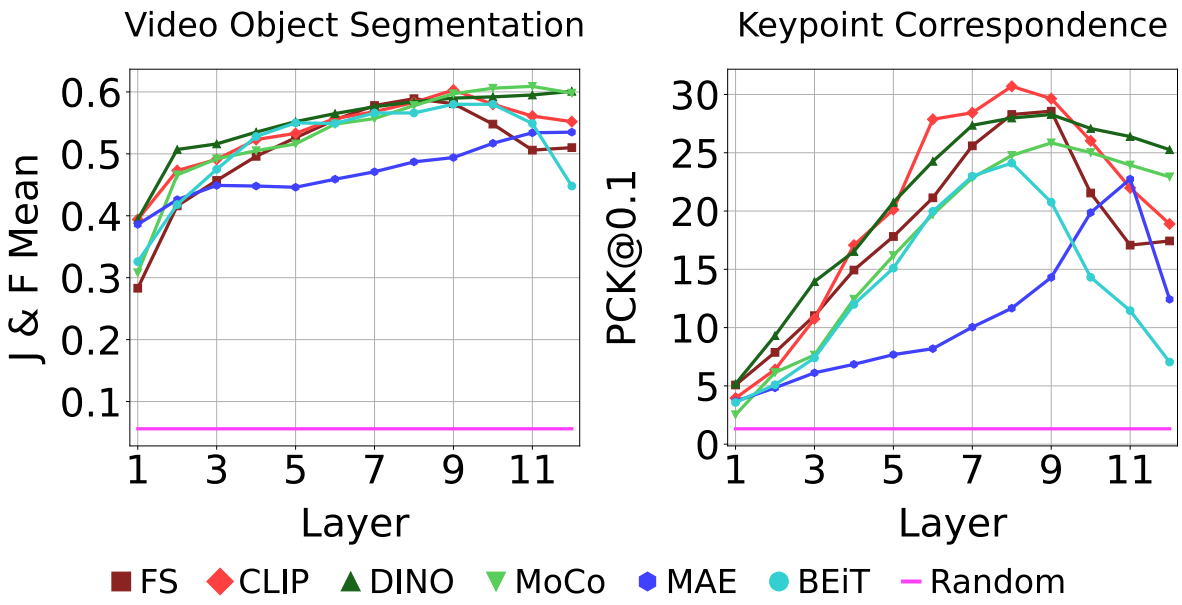


Figure 2.10: Local (pixel-level) downstream task analysis using the dense spatial token features. We perform DAVIS video segmentation (left) and SPair-71k keypoint correspondence (right).

2.5.2 Local Tasks

DAVIS Segmentation Propagation. DAVIS Segmentation Propagation is a dense prediction-based video localized task where frame-by-frame features are used to propagate the first frame segmentation mask to subsequent frames. Like Section 2.4.3, we use a tiling-based dense feature extraction strategy. Results are shown in Figure 2.10 (left). The contrastive techniques of DINO and MoCo perform the best while FS and CLIP, which are more image-level approaches, face a drop in performance towards the later layers. The local reconstruction-based methods, MAE and BEiT, are also much more competitive in this task. These results show that for purposes of constructing highly descriptive local features, contrastive methods like DINO and MoCo and reconstructive methods like BEiT can surpass features trained with explicit image-level supervision.

Keypoint Correspondence. We choose keypoint correspondence as an additional local-

Table 2.1: Best performance for each ViT on each downstream task with the corresponding best layer in parenthesis.

Model	**Task Performance (Best Performing Layer)**			
Dataset	ImageNet	ROxford5k (M)	Davis	SPair-71k
Metric	Top-1↑	mAP↑	J and F Mean↑	PCK@0.1↑
FS	**83.79 (12)**	**0.45 (12)**	0.59 (8)	28.56 (9)
CLIP	75.75 (12)	0.40 (12)	0.60 (9)	**30.70 (8)**
DINO	76.06 (12)	0.37 (12)	0.60 (12)	28.28 (9)
MoCo	71.59 (12)	0.31 (12)	**0.61 (11)**	25.85 (9)
MAE	45.19 (12)	0.15 (10)	0.54 (12)	22.74 (11)
BEiT	26.84 (8)	0.14 (8)	0.58 (9)	24.11 (8)
Random	0.10	0.02	0.06	1.32

focused downstream task. Given an image with annotated keypoints, the model must predict the position of corresponding keypoints in a paired image with similar content. We use the SPair-71k [72] dataset and follow the evaluation protocol of [55] and report the Percentage of Correct Keypoints (PCK) [80]. The results are summarized in Figure 2.10 (right). CLIP excels at this task, closely followed by both FS and DINO. Meanwhile, MoCo, BEiT, and MAE are all very competitive also. The position of the best layer varies significantly, from 8 for CLIP and BEiT to 11 for MAE.

2.5.3 Summary of Downstream Tasks

We summarize the best results for all downstream tasks in Table 2.1. We denote the best-performing layers in parenthesis. These results show that ViTs with different supervision methods **[1]** peak at different layers, and **[2]** perform best at different tasks. In image-level tasks like k-NN classification and retrieval, usually, the last layer works the best. For localized tasks like keypoint correspondence and video object segmentation, most models' peak performance happens a few

layers before the last one. This shows that always picking the last layer output is not optimal.

2.6 Discussion

In this work, we have performed an in-depth comparison of ViTs trained through different methodologies by examining their attention patterns, learned representations, and downstream task performance. We review some of the key findings of our analyses. First, different methods of supervision lead to ViTs that process local and global information in different orders. All ViTs have heads that align well with salient image content, but for the explicitly supervised models, the late-layer attention maps change into Sparse Repeating Patterns. In addition, all ViTs examined have learned to use Offset Local Attention Heads in multiple layers. While explicitly supervised ViTs have the most semantically rich representations at the image level, contrastive methods are competitive, and reconstruction-based methods can also learn meaningful CLS token representations even though said token is a placeholder and has no explicit supervision. Finally, there is no single best model for all the downstream tasks, and the best layer to extract representations from also varies greatly by task and model, so one should not simply take the last layer representation. ViTs have shown a great deal of potential, and we expect they will become more widely used in the coming years. We hope these insights can help with the future development of losses and architectures for Vision Transformers.

2.7 Appendix

In our primary results, we focused on comparing ViT-B/16 models trained with different supervision methods. In this Appendix, we present additional results for a wider range of ViT

variants including different architecture sizes and patch sizes. This analysis includes 20 models in total, summarized in Table 2.2. This Appendix is organized following the same three major domains: Attention, Features, and Downstream Tasks. We summarize all the key findings of our work in Table 2.3.

2.7.1 Additional Experimental Details

2.7.1.1 ViT Variants Examined

To provide a uniform platform for comparison, our primary results focus on ViT-B/16 models trained with six methods: Full Supervision, CLIP [62], DINO [26], MoCo-v3 [64], MAE [28], and BEiT [29]. In this Appendix, we present additional results examining different ViT variants for the above methods as provided by their original authors. In total, this expanded ViT collection contains 20 models including instances of ViT Small, Base, Large, and Huge as well as patch sizes 8, 14, 16, and 32. We continue to process all images at a 224×224 input resolution, meaning the number of spatial tokens for a given model will vary by patch size, from 47 spatial tokens for models with patch size 32 up to 784 spatial tokens for patch size 8. Overall, the models evaluated are summarized in Table 2.2. Note that the MoCo ViT-Small model is a modified variant with 12 heads per layer instead of 6.

2.7.1.2 ViT Training Details

In this section, we briefly outline the training protocols that were used to train each of the ViTs examined in this work. The CLIP, DINO, MoCo, MAE, and BEiT models are all official pre-trained models released by their original authors.

Table 2.2: Summary of all ViT Variants used in Appendix analysis. *MoCo S/16 uses 12 heads per layer instead of 6.

Model	Layers	Heads	Spatial Token Grid Size
FS S/32	12	6	7x7
FS S/16	12	6	14x14
FS B/32	12	12	7x7
FS B/16	12	12	14x14
FS B/8	12	12	28x28
FS L/16	24	16	14x14
CLIP B/32	12	12	7x7
CLIP B/16	12	12	14x14
CLIP L/14	24	16	16x16
DINO S/16	12	6	14x14
DINO S/8	12	6	28x28
DINO B/16	12	12	14x14
DINO B/8	12	12	28x28
MoCo S/16*	12	12	14x14
MoCo B/16	12	12	14x14
MAE B/16	12	12	14x14
MAE L/16	24	16	14x14
MAE H/14	32	16	16x16
BEiT B/16	12	12	14x14
BEiT L/16	24	16	14x14

Fully Supervised (FS). For FS, we work with models from the TIMM repository [61]. The FS models are pretrained on ImageNet21k and fine-tuned on ImageNet1k. The FS models are the only models in this study that are fine-tuned with ImageNet-1k labels. They are trained following the augmentation protocols of [60]. Specifically, the ViT-Base and Large models are trained using a combination of RandAugment [81] and Mixup [82], while the ViT-Small models use only RandAugment. All FS models also use weight decay [83].

CLIP. The goal of CLIP (**C**ontrastive **L**anguage-**I**mage **P**re-Training) is to train models with open-ended supervision provided by paired captions. The learning objective is simply to match images with their corresponding captions. CLIP models are joint vision and language models, which include separate encoder networks for the image and text inputs. For our analysis, we focus only on the properties of the visual encoding network. The authors pretrain the model

on 400M image-text pairs with a batch size of 32768 and mixed-precision to accelerate training and reduce memory usage. The only augmentation used is taking a random square crop from the resized image. They also use a cosine learning rate decay schedule.

MoCo. The **Mo**mentum **Co**ntrast (MoCo) method trains using contrastive learning with a momentum encoder, which is an exponential moving average of previous versions of the encoder. Under the contrastive objective, the encoder must generate representations for query image views that are similar to corresponding representations of key image views as generated by the momentum encoder. This strategy was proposed for CNNs and extended to ViTs. For MoCo v3 [64], the authors pretrain the model on ImageNet-1k without labels with a batch size of 4096. They follow a cosine learning rate decay. They use data augmentations like random resized cropping, horizontal flipping, color jittering, grayscale conversion, blurring, and solarization. They take two 224×224 crops for each image for each iteration.

DINO. The authors of DINO [26] describe their method as a form of self-**di**stillation with **no** labels. Their training strategy is based on MoCo [27] and they also use a momentum encoder, though they instead view their method as a student/teacher knowledge distillation framework. They pretrain the models on ImageNet-1k without labels with batch

Table 2.3: A comprehensive summary of the key observations of this work, including both the main paper and appendix results.

Key Observations	Analysis Methods	Sections	Figures & Tables
The attention maps of explicitly supervised ViTs devolve into Sparse Repeating Attention Patterns in the mid-to-late layers.	Average CLS Attention Maps	2.3.1	Figure 2.2
All ViTs studied learn to use Offset Local Attention Heads, suggesting they are fundamentally necessary in ViTs.	Aligned Aggregated Attention Maps	2.3.2	Figure 2.3
ViTs learn to process local and global information in different orders depending on their method of supervision.	Average Attention Distance	2.3.3	Figure 2.4
All ViTs studied differentiate salient foreground objects by the early-to-mid layers.	Attention Saliency IoU	2.3.4	Figure 2.5
Reconstruction-based self-supervised methods can learn semantically meaningful CLS representations, even when the CLS token is only a placeholder.	CKA Feature Similarity, Image Clustering by CLS Features	2.4.1, 2.4.2	Figures 2.6, 2.7
Supervised method's features are the most semantically rich, but contrastive self-supervised methods are comparable or even superior in some cases.	Image-, Object-, and Part-Level Feature Clustering	2.4.2, 2.4.3	Figures 2.7, 2.8
For localized tasks, the best performance often comes from a mid-to-late layer.	Local Downstream Tasks	2.5.2	Figure 2.10 Table 2.1
There is no single "best" training method or layer for all downstream tasks.	Local & Global Downstream Tasks	2.5.3	Figures 2.9, 2.10 Table 2.1
The positions of maximal activation in the Sparse Repeating Attention Patterns vary by input.	CLS Attention Maps	2.7.4.1	Figures 2.11-2.13
All models studied learn to use Offset Local Attention Heads, and some larger models even learn ones with diagonal offsets.	Aligned Aggregated Attention Maps	2.7.4.1	Figure 2.14
The order of local *vs*. global information processing in a ViT is primarily determined by the method of supervision and is largely unaffected by changes in architecture and patch size.	Average Attention Distance	2.7.4.2	Figure 2.15
For the expanded ViT collection, again all models differentiate salient foreground objects by the early-to-mid layers.	Attention Saliency IoU	2.7.4.3	Figure 2.16
Explicitly supervised ViTs with patch size 32 are less impacted by Sparse Repeating Attention Patterns, suggesting they may be an indication of overfitting.	Averaged CLS Attention Maps	2.7.4.1, 2.7.4.3	Figure 2.17
The last layer spatial representations of self-supervised methods are similar across changes in architecture size and patch size, but this is not consistently true for explicitly supervised methods.	CKA Feature Similarity	2.7.5.2	Figure 2.19
Both MAE and BEiT show X patterns in their depth-wise feature CKAs, suggesting an encoder/decoder internal structure.	CKA Feature Similarity	2.7.5.3	Figures 2.21, 2.22
For larger MAE ViTs, the later layers appear to act more like decoder layers, even thought MAE has a separate decoder.	CKA Feature Similarity	2.7.5.3	Figure 2.21
Residual connection analysis provides further evidence of a fundamental shift in information processing in the mid-to-late layers of explicitly supervised ViTs.	Residual Connection Analysis	2.7.5.4	Figure 2.23
BEiT L/16 learns extremely expressive part-level features compared with BEiT B/16.	Part-Level Feature Clustering	2.7.5.5	Figure 2.27
Larger architectures tend to give better feature quality and downstream performance.	Feature Clustering, Local & Global Downstream Tasks	2.7.5.5, 2.7.6.2	Figures 2.24-2.31
ViTs with smaller patch sizes unsurprisingly perform better at localized downstream tasks.	Local Downstream Tasks	2.7.6.2	Figures 2.30, 2.31
Reconstruction-based ViTs show the largest variance in their performance characteristics on downstream tasks.	Local & Global Downstream Tasks	2.7.6.2	Figures 2.28-2.31
For the expanded group of ViTs, once again there is no single "best" training method or layer for all downstream tasks.	Local & Global Downstream Tasks	2.7.6.2	Figures 2.28-2.31 Table 2.5

size 1024. They follow a cosine learning rate and weight decay. They use data augmentations like color jittering, gaussian blur, and solarization similar to BYOL [84]. Multi-crop [85] is also used.

MAE. The **M**asked **A**uto**e**ncoder (MAE) [28] method is a reconstruction-based training objective where a large portion of input patches/tokens are masked out. The rational of MAE is that, because a large percentage of the image content is missing, the network must learn representations that embed meaningful high-level semantics to reconstruct the missing regions. MAE uses both an encoder and decoder network, though the decoder is discarded after pretraining. The authors pretrain the model on ImageNet-1k without labels with a batch size of 4096. They do not use color jittering, drop path or gradient clipping and only apply random resized crop augmentation. They use a masking ratio of 0.75 which also improves the efficiency of training by significantly decreasing the token count in the encoder. They also use a cosine learning rate decay schedule.

BEiT. BEiT [29] stands for **B**idirectional **E**ncoder representation from **I**mage **T**ransformers, and it is based on BERT [86], a well-known masked reconstruction learning method for NLP. In contrast to MAE, BEiT does not perform pixel-level reconstruction, but instead uses a tokenizer to convert image patches into discrete tokens. The BEiT learning objective is to predict the token values for the masked patches. Unlike MAE, BEiT does not include a separate decoder network. BEiT is trained with a masking ratio of roughly 0.4, though they also employ a block-masking method which masks out larger adjacent groups of tokens. They pretrain BEiT on ImageNet-1k with a batch size of 2048, and they include random resized cropping, horizontal flipping, and color jittering augmentations. They also utilize cosine learning rate decay. Note that the authors have provided both BEiT models before and after fine-tuning with ImageNet labels. For our anal-

ysis, we work with the non-fine-tuned versions, in order to focus on just the effects of the BEiT pretraining method.

For more details and exact parameters, please refer to the corresponding papers and codebases for each of the models.

2.7.2 Random Chance Scores

During our Feature Clustering and Downstream Task Analysis, we present random chance scores for both the clustering metrics and downstream task scores. To evaluate these scores, we repeat the task analysis replacing all ViT features with uniformly distributed Gaussian noise. To be more specific, we generate arrays of Gaussian noise with the exact same dimensions as the extracted feature arrays of a ViT B/16 model. These random chance scores are heavily influenced by the underlying data. For example, we see that the random chance score is quite high for the object-level clustering purity scores on COCO. This can be attributed to the dataset's highly imbalanced object distribution. Still, this method of random chance evaluation is informative as it effectively acts as a baseline model where all feature vectors contain absolutely no useful information.

2.7.3 Dense Feature Extraction

Certain local tasks, like object segmentation, benefit from having a denser array of high quality features. For an input image of size 224×224, a ViT with patch size 16 produces a feature array with size 14×14. This low resolution can be very limiting for localized tasks, like DAVIS Video Segmentation Propagation. While some of the ViTs evaluated do support variable

input size, others are hard-coded to operate at a fixed size. To generate a denser feature grid while also providing a level playing field, we propose a simple dense feature extraction strategy using image tiling.

We begin by rescaling the smaller image dimension to size 448 (twice the input resolution). We then scale the larger image dimension to the nearest integer multiple of the model patch size, in order to preserve the image aspect ratio as best as possible. Finally we slice the image into non-overlapping tiles of size 224×224. Then we extract ViT features for each of the tiles and concatenate the features together. Unless the image is exactly square, this leaves some leftover image content along the larger image dimension. For these areas, we take two additional crops which do overlap other image tiles, but for these areas the features are discarded, while the non-overlapping features are concatenated to the rest. The final product is a feature array that is twice as dense as the original while also respecting the original image aspect ratio.

In Table 2.4, we present an ablative analysis comparing the DAVIS Video Segmentation performance of all models with and without dense feature extraction enabled. All models see a significant performance boost with dense feature extraction, especially models with patch size 32.

2.7.4 Attention Analysis

2.7.4.1 Expanded Attention Visualizations

Figure 2.11 and Figure 2.12 provide additional visualizations of CLS token attention maps in ViT-B/16 models for single input images. These visualizations show all layers (rows) and all heads (columns). From these views, we can see clear signs of the Sparse Repeating Attention

Table 2.4: Comparison of Dense vs. Normal feature extraction for the DAVIS Video Object Segmentation task. Results show for the best layer per model, with layer number in parenthesis.

Model	J and F Mean	
	Normal	Dense
FS S/32	0.18 (12)	0.39 (9)
FS S/16	0.34 (10)	0.58 (8)
FS B/32	0.17 (10)	0.38 (9)
FS B/16	0.34 (10)	0.59 (8)
FS B/8	0.51 (9)	0.68 (9)
FS L/16	0.34 (19)	0.56 (13)
CLIP B/32	0.19 (12)	0.41 (9)
CLIP B/16	0.35 (9)	0.60 (9)
CLIP L/14	0.38 (17)	0.60 (17)
DINO S/16	0.32 (11)	0.61 (11)
DINO S/8	0.52 (11)	0.73 (12)
DINO B/16	0.32 (12)	0.60 (12)
DINO B/8	0.51 (11)	0.73 (10)
MoCo S/16	0.34 (11)	0.6 (10)
MoCo B/16	0.33 (12)	0.61 (11)
MAE B/16	0.29 (12)	0.54 (12)
MAE L/16	0.31 (24)	0.55 (23)
MAE H/14	0.36 (30)	0.59 (30)
BEiT B/16	0.31 (7)	0.58 (9)
BEiT L/16	0.36 (17)	0.61 (15)

Patterns in the mid-to-late layers of the FS and CLIP models. These patterns are strongly repeated across the head and layer axes. Note that the specific token positions that give strong activations are different for the different input images.

Figure 2.13 shows one ViT attention map per model for 10 sample images over a wide array of ViT variants. The final row displays the average attention over 5000 images. The head selected is the first head of the final layer of each ViT. For the explicitly supervised models, FS and CLIP, we again see mainly Sparse Repeating Attention Patterns. However, for the models with patch size 32, we also see some attention on object-centric regions. This holds true for FS S/32, FS B/32, and CLIP B/32. Because these models use larger patches, their token grids are a quarter of the size, making these models four times narrower than the models with patch size 16. The fact that Sparse Repeating Attention Patterns do not emerge as strongly for these

smaller models may suggest that they are an indicator of overfitting in ViTs. For the other FS and CLIP models, we sometimes see faint traces of salient objects highlight in the attention maps, but this occurs alongside the Sparse Repeating Attention Patterns. For DINO, MoCo, and MAE, all models produce attention maps that tend to align well with the salient object. For BEiT, the attention maps do not correlate well. As we previously noted, the final layers of BEiT must serve as a built-in decoder, which may explain why its final layers are dissimilar to DINO, MoCo, and MAE.

We find that every model learns instances of Offset Local Attention Heads, and some larger models even have ones with a diagonal offset. We present one example per model in Figure 2.14, but be aware that all models have many Offset Local Attention Heads with different offsets. For the explicitly and contrastively supervised models, Offset Local Attention Heads typically only occur in the first 3 to 6 layers, but for the reconstruction-based models, MAE and BEiT, we can find them in deeper layers too.

To provide the reader a complete view of the size and number of attention heads in each ViT, we present two plots that visualize all layers and all heads of all 20 ViTs. Figure 2.17 presents the average CLS token attention over 5000 sample images. When viewed this way, it is clear how widespread the Sparse Repeating Activation Patterns are over all of the mid-to-later layer heads of the FS and CLIP models. For FS S/32 and B/32 we can see clear signs of Sparse Repeating Activation Patterns, but for CLIP B/32 we instead see far more centered circular blobs, similar to those we observe in the later layers of DINO and MoCo. We also note that some of the early-layer heads (layers 1-3) of the DINO models produce semi-repetitive grid-like patterns that somewhat resemble the Sparse Repeating Attention Patterns seen in CLIP and FS. However, on closer inspection, we believe these heads represent a different phenomenon. The attention

patterns in these layers have more variations across heads and layers, and they are not identically repeated as is seen in the FS and CLIP models. Also, these heads come in the early layers, not the mid-to-late layers. For this reason, we hypothesize these heads are learning to extract an initial sparse down-sampling of the image, which would be especially beneficial for the DINO models with patch size 8 due to their much larger token counts. Finally, Figure 2.18 shows the Aligned Aggregated Attention Maps for all the spatial tokens. This view highlights the great variety of local attention heads used in each ViT. These figures are best viewed digitally and in color.

2.7.4.2 Attention Distance for ViT Variants

In Figure 2.15 we present the Average Attention Distances per-layer for the full ViT collection, broken out by supervision type. For our distance computations, we have normalized the distances such that the token grids are within a 1×1 square, which allows us to compare models with different patch sizes and hence different token grid sizes. We see that the trends of local-vs-global processing order is consistent within supervision groups. For FS, CLIP, DINO, and MoCo we again see an intial high distance, a dip to lower distances, and an increase again in the later layers. Meanwhile, for MAE we again see lower attention distance in the later layers. This result shows that the order of local-vs-global information processing in a ViT is primarily impacted by the method of supervision and is largely unaffected by changes in architecture size and patch size.

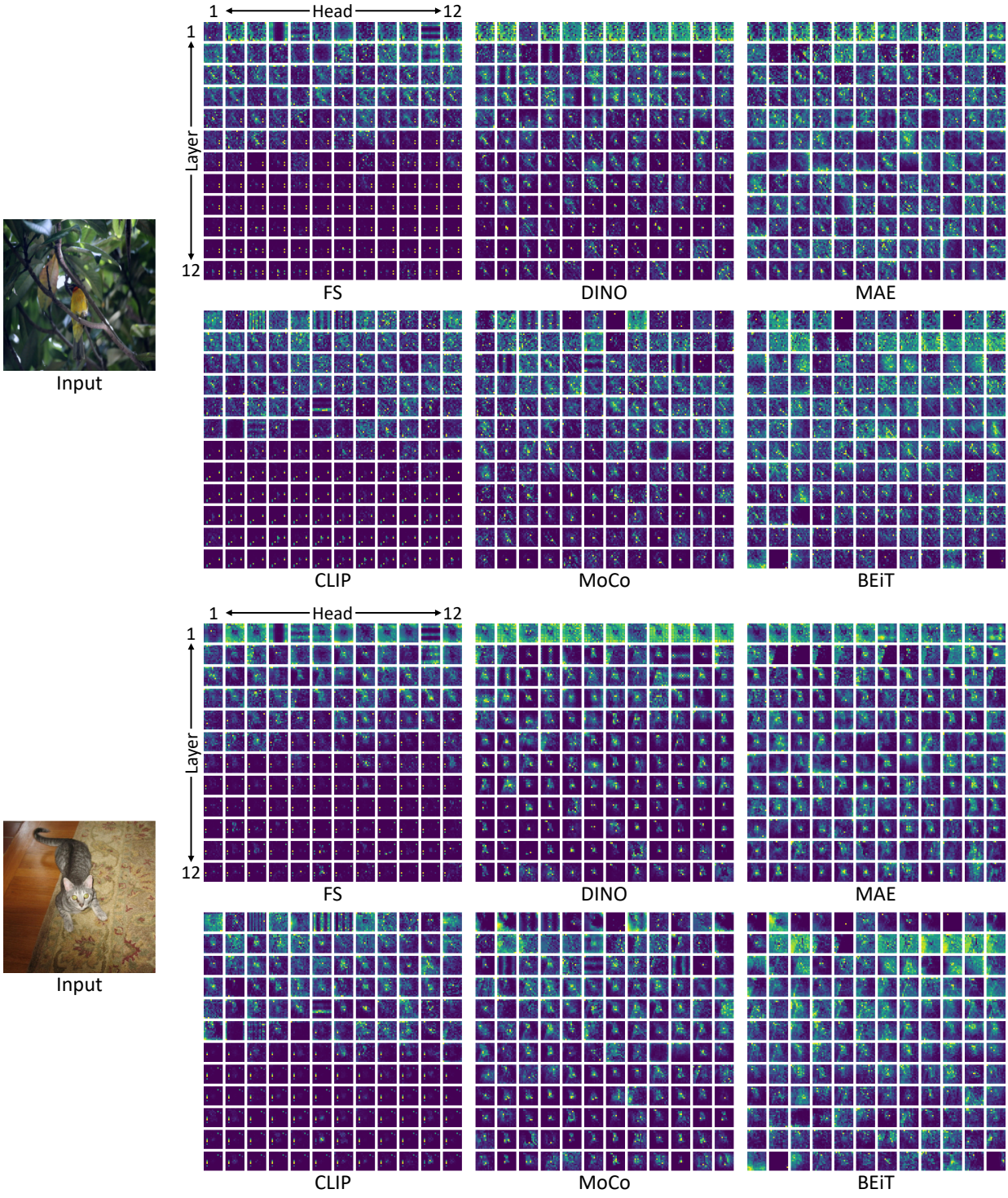


Figure 2.11: Visualizing all CLS token attention maps for all heads and all layers in ViT B/16 models for single input images (left). The FS and CLIP models show Sparse Repeating Attention Patterns in the mid-to-late layers, where a small group of spatial token positions at seemingly arbitrary positions have strong and consistent activations shared across both heads and layers. Note that the positions of these strong repetitive activations are different for the two inputs.

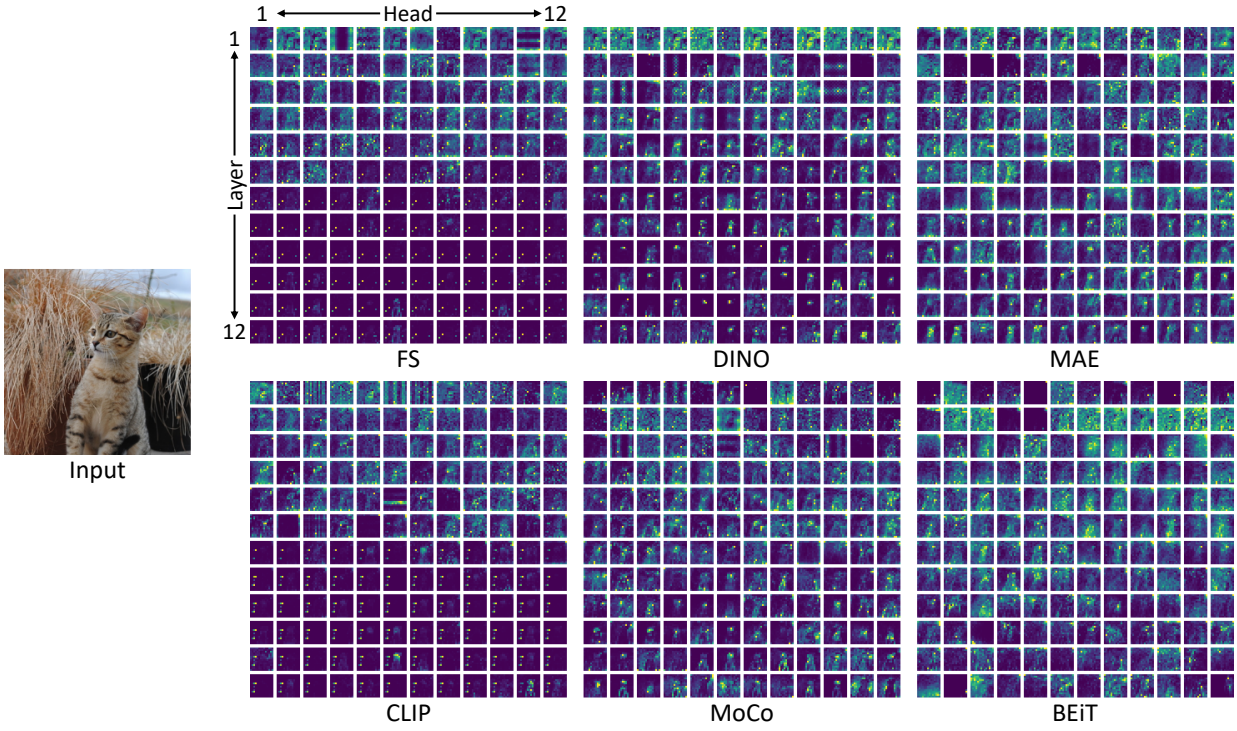


Figure 2.12: Visualizing all CLS token attention maps for all heads and all layers in ViT B/16 models for a single input image (left).

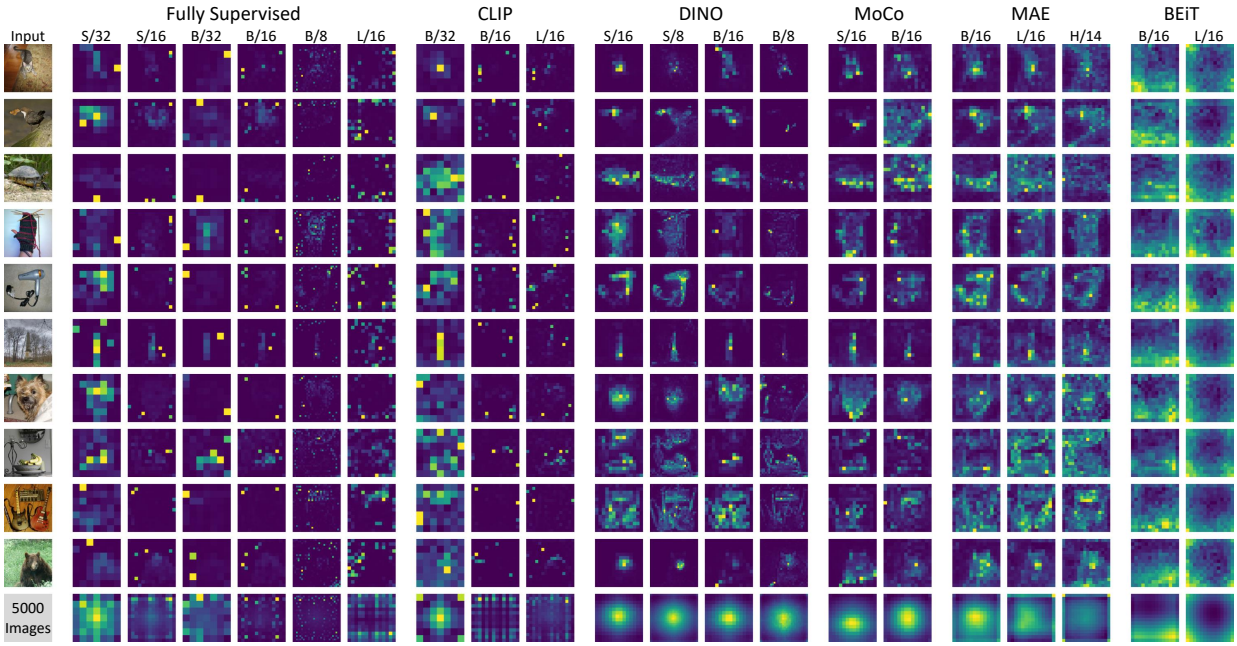


Figure 2.13: **Sample CLS token attention maps for a wide range of ViT variants.** For each ViT and input image, we show the the attention map of the CLS token of the first head of the final layer. The bottom row shows the averaged activation over 5000 ImageNet images.

2.7.4.3 Attention Salience for ViT Variants

In this section, we present additional experimental details for our Attentional Saliency Analysis, followed by results for the full ViT collection. In addition to PartImageNet [67], we also performed this analysis with COCO [68], however the results are extremely similar for the two datasets. The PartImageNet dataset contains 11 superclasses, whose members contain similar part structures (biped, quadruped, car). When sampling from PartImageNet, we take 500 samples per superclass, or all samples for ones with less than 500. Within superclasses, we evenly sample from each of the subclasses, or if a subclass is fully sampled we continue to sample evenly from the remaining classes. This yields a mostly balanced collection of 5294 images. PartImageNet divides different subclasses into the train, validation, and test partitions, but for our analysis we work with all three partitions together. For COCO, we simply sample the first 5000 images of the 2017 validation set.

Figure 2.16 summarizes the results for both PartImageNet and COCO with both CLS token attention and average spatial token attention. The patterns of scores are very similar for PartImageNet and COCO. We see that the explicitly supervised methods again face a decrease in IoU in the mid-to-late layers with the emergence of Sparse Repeating Attention Patterns. This is with the exception of CLIP B/32 which has a much better IoU in the later layers. This result matches Section 2.7.4.1, where we observed that CLIP B/32 is less impacted by the Sparse Repeating Attention Patterns. For the other self-supervised methods, we see that the same general trends hold, usually with slightly higher IoUs from larger models or models with smaller patches.

2.7.5 Feature Analysis

2.7.5.1 The CKA metric

In our Feature Analysis section, we compare learned representations through Centered Kernel Alignment (CKA) [76, 77], which is able to align and rescale neural features to enable similarity measurements. Specifically, we used batched CKA [78], which can be represented as follows:

$$CKA_{batched} = \frac{\frac{1}{k}\sum_{i=1}^{k} HSIC_1(\mathbf{X}_i\mathbf{X}_i^T, \mathbf{Y}_i\mathbf{Y}_i^T)}{\left(\sqrt{\frac{1}{k}\sum_{i=1}^{k} HSIC_1(\mathbf{X}_i\mathbf{X}_i^T, \mathbf{X}_i\mathbf{X}_i^T)} * \sqrt{\frac{1}{k}\sum_{i=1}^{k} HSIC_1(\mathbf{Y}_i\mathbf{Y}_i^T, \mathbf{Y}_i\mathbf{Y}_i^T)}\right)}, \tag{2.1}$$

where X_i and Y_i are the feature representation matrices of the i^{th} batch from the two models, k is the number of batches and $HSIC_1$ is as follows:

$\text{HSIC}_1(\mathbf{K}, \mathbf{L}) = \frac{1}{n(n-3)}\left(tr(\hat{\mathbf{K}}\hat{\mathbf{L}}) + \frac{\mathbf{1^T\hat{K}11^T\hat{L}1}}{(n-1)(n-2)} - \frac{2}{n-2}\mathbf{1^T\hat{K}\hat{L}1}\right)$, where $\hat{\mathbf{K}}$ and $\hat{\mathbf{L}}$ are $\mathbf{K}, \mathbf{L}$ with diagonals set to 0 and n is the batch size. We use the implementation from Subramanian [87] for our analysis.

2.7.5.2 Last Layer Comparisons for ViT Variants

In the main paper we analyzed the last layer CKA between the CLS and spatial tokens across the B/16 models separately. Here we expand our analysis to the wider collection of ViT variants. As can be seen from Figure 2.19 (left), for the CLS tokens, similar supervision strategies create similar representations. Groups emerge with DINO and MoCo forming one subset while MAE and BEiT form another. The FS and CLIP models form their own sub-groups. Some of the

FS models also show comparatively high similarity with MoCo and DINO models. Again we see that the MAE CLS representations have a moderate similarity with explicitly and constrastively supervised methods.

From Figure 2.19 (right), we see that the internal similarity within the FS and CLIP groups are more fragmented. Meanwhile, the self-supervised models show more consistency, having higher spatial feature similarity within and between self-supervision methods. DINO and MoCo show very high similarity amongst themselves due to their similar training methods. The MAE spatial features also show high similarity to those of DINO and MoCo. BEiT shows comparatively high similarity with these other self-supervised methods. FS and CLIP are not too similar with each other or with the other models with the exception of CLIP B/32 and a few FS models like B/16 and B/8 which show comparatively high similarity with DINO and MoCo models. This separation of CLIP and FS can be attributed to the supervision which is applied to only the CLS token, which may make their final layer spatial representations less consistent.

2.7.5.3 Depth-Wise CKA Analysis

Self-Comparison of ViT B/16 Models. Figure 2.20 shows the CKA plots across multiple layers of the same model for different training methods. For brevity and consistency we focus on the ViT B/16 models. For all the CKA plots we use the features from the batch norm layers due to their well-behaved outputs. We see that the different models show variations in the development of information. For FS, the final layers have a lower similarity to the earlier layers, as compared with CLIP, DINO, and MoCo. For MAE, we see two distinct blocks of similarity divided around the middle layer. For BEiT, we see a clear X pattern, which we analyze more in a subsequent

section.

Going deeper into MAEs. Figure 2.21 shows the CKA plot for MAE models Base, Large and Huge from left to right. It can be seen that as we move from a smaller model to a larger model (for example from Base to Large), the bottom left quadrant of larger models CKA matches the full CKA for the smaller model. This indicates that a larger model in this case encodes information in a similar way as the smaller model in its initial layers but ends up having more specialized later layers at the end. A similar trend can be observed when going from Large to Huge.

X pattern for MAE and BEiT. As shown in Figure 2.21 and Figure 2.22, MAE and BEiT show an X-like pattern in their CKA plot (with the exception of MAE B/16). This indicates that the late layer features of these models are similar to the early layers but not the middle layers. We hypothesize that this is due to the reconstruction-oriented nature of the training losses, showing that in their final layers MAE and BEiT are trying to recreate the same local information that is present in the initial layers. It should be noted that this X-pattern arises for all sizes of BEiT, but not in MAE B/16. We attribute this to the fact that MAE has a separate decoder module which is discarded after training, while BEiT does not have a separate decoder. This means that BEiT needs to inherently learn a decoder which leads to emergence of this X pattern at both sizes. For MAE, the fact that the X pattern emerges more clearly for the larger ViT variants suggests that the additional layers of these models start specializing for the task of decoding. This has important implications for the MAE training method, as it suggests that, for larger MAEs, late layers learn to act in a more decoder-like way, which may limit the usefulness of these layers for downstream tasks.

2.7.5.4 Residual Connection Analysis

Previously works [16] have contrasted CNNs and ViTs by comparing the features propagated through the skip connections and normal connections. We extend this analysis to ViTs trained with varying supervision techniques. Figure 2.23 shows the CKA between the features coming from the skip connection (Y-axis) and the normal pathway (X-axis) for the MHA layer of each block. For CLIP and FS we can see a similar trend, initially the skip connection carries similar information as the normal path but after a certain point the information it carries becomes very different (dark regions). This also correlates with the emergence of the Sparse Repeating Patterns observed in Section 2.7.4.1, providing further evidence that a fundamental shift in information processing behavior occurs in the mid-to-late layers of explicitly supervised ViTs. For MoCo and DINO, this shift in behavior does not happen and as the depth increases the skip connections and normal pathways still have similar representations. MAE is another special case where, given the reconstruction nature of the loss, at multiple depth locations the normal pathway representation is similar to the skip connection representation. For BEiT, there is again an X-like pattern, likely because it needs to start reconstructing the complete input. MAE does not show an X-like pattern, despite its similar reconstruction objective. Again we theorize that this occurs because MAE has a separate decoder that is discarded after training.

2.7.5.5 Additional Clustering Analysis

In this section, we expand our feature clustering analysis to the full collection of models. In addition to Cluster Purity, we also report results for Normalized Mutual Information (NMI) and Adjusted Random Index (ARI). We see similar trends for all three clustering metrics.

Image-Level CLS Feature Clustering Results shown in Figure 2.24. For FS, CLIP, DINO, and MoCo the same general trends hold. Cluster quality rises faster for DINO and MoCo, but FS and CLIP catch up rapidly and overtake at the end. For the deeper model variants (FS L/16 and CLIP L/14) the trends are consistent when plotted against normalized block depth, meaning that semantic information actually emerges half as quickly. For MAE, the larger model variants lead to significantly better cluster purity, which also rises earlier as the models get larger. In contrast, for BEiT cluster purity is generally worse for the larger L/16 model.

Image-Level Spatial Feature Clustering Results shown in Figure 2.25. For FS, CLIP, DINO, and MoCo the same general trends show, though with larger models tending to do slightly better. Interestingly, the spatial features of BEiT L/16 have an increase in cluster purity, which is surprising as its CLS tokens saw a decrease in the previous section.

Object-Level Spatial Feature Clustering Results shown in Figure 2.26. For FS, CLIP, DINO, and MoCo again the general trends hold, with larger models or those with smaller patch size doing slightly better. However, for FS the L/16 variant did worse than B/16. For FS the best scores are achieved by the B/8 variant. Like the previous section, we see a significant boost for BEiT L/16 over B/16.

Part-Level Spatial Feature Clustering Results shown in Figure 2.27. For part-level feature clustering, we previously observed that, for the B/16 models, the self-supervised methods are much more competitive with the explicitly supervised methods. This trend still holds here, with BEiT L/16 performing particularly well, seeing a huge boost over BEiT B/16. For all metrics, BEiT L/16 is on par with the best explicitly and constrastively supervised methods. Larger models and those with smaller patch size generally provide better part-level feature clusters, with the

Table 2.5: Best performance for each ViT on each downstream task with the corresponding best layer in parenthesis.

Model		**Task Performance (Best Performing Layer)**											
Dataset	Layers	ImageNet		ROxford5k		RParis6k		DAVIS			SPair-71k		
Metric		Top-1↑	Top-5↑	mAP↑ (M)	mAP↑ (H)	mAP↑ (M)	mAP↑ (H)	J Mean↑	F Mean↑	J and F Mean↑	PCK@0.1↑	PCK@0.05↑	PCK@0.01↑
FS S/32	12	74.48 (12)	90.33 (12)	0.33 (12)	0.12 (12)	0.63 (12)	0.38 (12)	0.43 (8)	0.35 (9)	0.39 (9)	15.28 (8)	4.34 (8)	0.13 (8)
FS S/16	12	80.64 (12)	93.71 (12)	0.34 (12)	0.1 (11)	0.66 (12)	0.40 (12)	0.57 (8)	0.60 (9)	0.58 (8)	26.37 (8)	11.62 (8)	0.68 (9)
FS B/32	12	79.08 (12)	92.8 (12)	0.33 (12)	0.12 (12)	0.67 (12)	0.43 (12)	0.42 (9)	0.34 (9)	0.38 (9)	15.49 (9)	4.19 (8)	0.15 (7)
FS B/16	12	83.79 (12)	95.01 (12)	0.45 (12)	0.19 (12)	0.72 (12)	0.51 (12)	0.58 (8)	0.60 (8)	0.59 (8)	28.56 (9)	12.33 (8)	0.63 (7)
FS B/8	12	85.58 (12)	95.71 (12)	0.45 (12)	0.16 (12)	0.73 (12)	0.51 (12)	0.66 (7)	0.70 (9)	0.68 (9)	36.09 (9)	21.97 (9)	1.61 (9)
FS L/16	24	85.03 (24)	95.39 (24)	0.42 (24)	0.15 (24)	0.73 (24)	0.51 (24)	0.56 (13)	0.57 (13)	0.56 (13)	30.99 (17)	13.49 (15)	0.79 (14)
CLIP B/32	12	70.90 (12)	89.54 (12)	0.38 (12)	0.10 (12)	0.66 (12)	0.41 (12)	0.44 (9)	0.37 (9)	0.41 (9)	18.55 (8)	5.37 (8)	0.26 (8)
CLIP B/16	12	75.75 (12)	92.27 (12)	0.40 (12)	0.11 (12)	0.71 (12)	0.48 (12)	0.58 (9)	0.62 (9)	0.60 (9)	30.70 (8)	13.61 (8)	0.98 (6)
CLIP L/14	24	80.24 (24)	94.15 (24)	0.45 (24)	0.17 (24)	0.70 (24)	0.49 (24)	0.57 (14)	0.62 (17)	0.60 (17)	36.04 (15)	16.72 (15)	1.15 (13)
DINO S/16	12	74.61 (12)	90.08 (12)	0.38 (12)	0.14 (12)	0.61 (12)	0.33 (12)	0.60 (11)	0.63 (11)	0.61 (11)	26.72 (9)	11.68 (9)	0.55 (9)
DINO S/8	12	74.34 (12)	90.15 (12)	0.36 (12)	0.12 (12)	0.59 (12)	0.30 (12)	0.70 (12)	0.77 (12)	0.73 (12)	31.14 (8)	18.15 (8)	1.48 (8)
DINO B/16	12	76.06 (12)	91.40 (12)	0.37 (12)	0.11 (12)	0.62 (12)	0.35 (12)	0.59 (12)	0.61 (12)	0.60 (12)	28.28 (9)	12.00 (7)	0.65 (6)
DINO B/8	12	77.70 (12)	92.24 (12)	0.40 (12)	0.13 (11)	0.65 (12)	0.37 (12)	0.69 (10)	0.77 (10)	0.73 (10)	33.17 (8)	19.04 (8)	1.66 (5)
MoCo S/16	12	68.71 (12)	86.36 (12)	0.27 (12)	0.07 (12)	0.50 (12)	0.22 (12)	0.58 (10)	0.62 (10)	0.6 (10)	24.88 (9)	10.92 (9)	0.39 (11)
MoCo B/16	12	71.59 (12)	88.37 (12)	0.31 (12)	0.08 (12)	0.51 (12)	0.22 (12)	0.59 (11)	0.62 (11)	0.61 (11)	25.85 (9)	10.64 (8)	0.43 (10)
MAE B/16	12	45.19 (12)	65.32 (12)	0.15 (10)	0.02 (10)	0.28 (10)	0.08 (10)	0.54 (11)	0.54 (12)	0.54 (12)	22.65 (11)	10.59 (11)	0.44 (11)
MAE L/16	24	60.80 (20)	78.9 (20)	0.19 (21)	0.03 (21)	0.35 (21)	0.11 (21)	0.55 (23)	0.56 (23)	0.55 (23)	27.65 (19)	13.02 (22)	0.60 (21)
MAE H/14	32	63.16 (23)	79.87 (23)	0.20 (23)	0.03 (30)	0.39 (23)	0.13 (23)	0.58 (31)	0.61 (30)	0.59 (30)	27.50 (26)	13.65 (26)	1.35 (26)
BEiT B/16	12	26.84 (8)	45.12 (8)	0.14 (8)	0.02 (8)	0.20 (8)	0.05 (10)	0.57 (10)	0.59 (9)	0.58 (9)	24.11 (8)	11.02 (8)	0.54 (7)
BEiT L/16	24	41.24 (18)	62.79 (18)	0.16 (18)	0.02 (18)	0.25 (17)	0.06 (17)	0.58 (17)	0.64 (15)	0.61 (15)	37.52 (15)	18.22 (16)	1.04 (16)
Random	-	0.10	0.49	0.02	0.01	0.04	0.03	0.03	0.08	0.06	1.32	0.34	0.02

exception of MAE, where the large models actually do worse.

2.7.6 Downstream Task Analysis

2.7.6.1 Keypoint Correspondence Additional Details

As part of our Downstream Task Analysis, we present results for Keypoint Correspondence as an additional local-focused task. Given an input image with a set of human annotated keypoints, a model must predict the position of corresponding keypoints in a second paired image with the same type of object. Challenges in this task include changes in scale, size, and large intraclass variations. Correspondence is a prerequisite step in applications such as pose estimation [88], 3D reconstruction [88], and edit propagation in images and videos [89]. We use the SPair-71k [72] dataset consisting of 1800 images from 18 categories. Following the evaluation protocol used by Amir et al. [55], we randomly sample 20 image pairs from each category of the

test set and compute PCK [80] (percentage of correct keypoints) for each of the 18 categories. Given the dense ViT spatial token features of a source image, a target image, and source keypoint, we (1) get the corresponding feature vector of the keypoint in the source image, (2) find the nearest neighbor of this feature vector in the target image, and (3) get the 2D location of the nearest neighbor in the target image. The keypoint prediction is considered correct if it is within, a threshold $\alpha \cdot \max(H, W)$ of the groundtruth correspondence, where α is a constant and (H, W) are the height and width of the target image. We report the average PCK@0.1, PCK@0.05, and PCK@0.01.

2.7.6.2 Results for ViT Variants

ImageNet Classification As seen in Figure 2.28, the general trends as reported in the main paper for this task hold for each training method. The FS B/8 has the highest Top-1 and Top-5 performance. For FS, CLIP, DINO and MoCo the trends show that performance improves as we go to later layers. It is also interesting to see that under normalized depth, the larger MAE models peak earlier than the smaller ones and show a higher peak performance. For BEiT, the peak performance occurs at a similar relative depth but is higher for the L/16 model.

Image Retrieval As shown in Figure 2.29, the Base and Large models for FS and CLIP perform well for this image-level task. This aligns with our observations in the main paper. The FS B/16 performs the best on ROxford5k while the FS B/8 and L/16 are the best performers on RParis6k. For the FS, CLIP, MoCo and DINO models, performance improves as we go to later layers in most cases. MAE and BEiT again peak early in the mid-to-late layers. The general observations and trends for this task are similar to the k-NN task as they are both image-level global tasks.

The only difference being that, for this task, the later layers of FS and CLIP show a sudden improvement while the earlier layers are flatter when compared to the trends for k-NN.

DAVIS Segmentation Propagation As shown in Figure 2.30 many models peak at an earlier layer. The best performance comes from DINO B/8 and S/8, followed closely by FS B/8. Meanwhile, the models with patch size 32 see a significant drop in performance. Given the dense-prediction-based nature of this task, these methods which are trained with a smaller patch size have finer features which give them a boost in performance. The reconstruction-based models, BEiT and MAE, are also very competitive in this task, performing on par with FS, CLIP, DINO, and MoCo among the models with patch size 16.

Keypoint Correspondence We show comparisons on this task in Figure 2.31. It should be highlighted that BEiT L/16 performs the best for PCK@0.1, again showing a massive improvement over BEiT B/16. This large boost correlates with the its improved part-level feature purity observed in Section 2.7.5.5. In general the smaller patch size models like DINO B/8, DINO S/8, CLIP L/14 and FS B/8 are also good performers. Due to their finer feature grids, these methods perform much better at the stricter thresholds (PCK@0.05 and PCK@0.01). All models on this task peak around mid-to-late layers.

2.7.6.3 Summary of Downstream Tasks

We report the best result for each model along with the layer at which it occurs in Table 2.5. This table captures all downstream tasks and summarizes all metrics for each task. We would like to highlight that **[1]** different models peak at different layers, based on type of task, local vs. global, and **[2]** no one model is the best model for all tasks.

2.7.6.4 ImageNet-1k Classification with Linear Probes

We present additional results for ImageNet classification with Linear Probes in Table 2.6. For each model, we trained a linear layer on the last layer features for 20 epochs on the ImageNet-1k training set, and report results on the validation set. For BEiT we instead use layer 8, which gave the best k-NN classification results. This analysis includes variations based on protocols from the compared works. We present results for CLS token features in row 2 and average-pooled spatial token features (proposed by BEiT) in row 3. We also test the addition of a batch normalization layer before the linear layer (proposed by MAE) in rows 4 & 5. As FS is trained with ImageNet labels, it performs best in all settings. For approaches with explicit CLS supervision (FS, CLIP, DINO, MoCo), the CLS token features give higher accuracy. For MAE and BEiT, due to the local nature of their supervision, their spatial features give better performance. Batchnorm is generally beneficial for all models and features.

Table 2.6: Accuracy@1 of ViT-B/16 models for Linear Probing on ImageNet-1k val. *required reduced LR for stable training.

Feature	FS	CLIP	DINO	MoCo	MAE	BEiT
CLS	83.86	65.63	73.03	74.26	49.52	9.87*
Spat.	82.31	52.53	37.37	62.47	52.01	29.81
CLS+BN	**84.40**	**78.63**	**76.39**	**74.51**	59.31	41.68
Spat.+BN	82.83	74.59	68.09	68.58	**59.86**	**44.27**

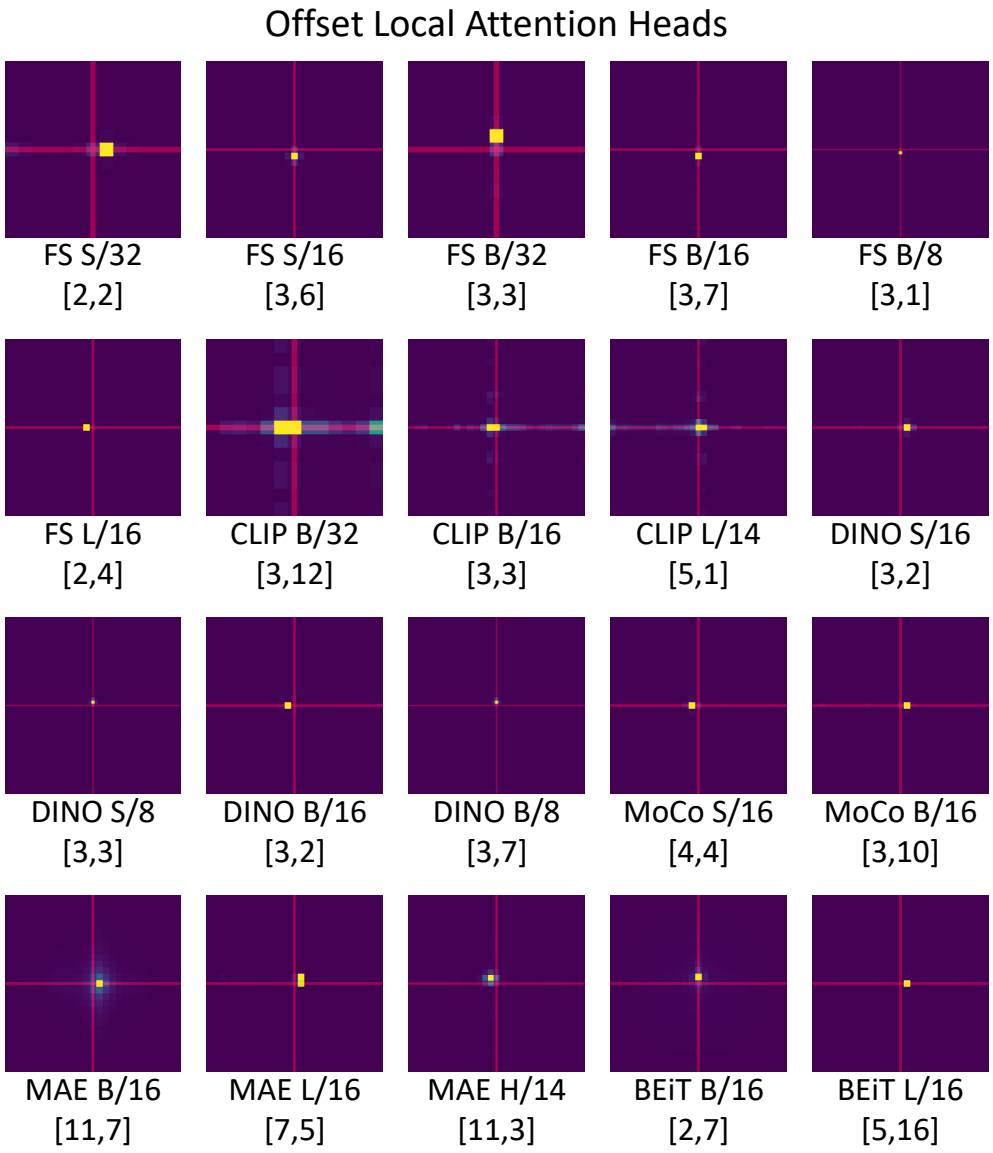


Figure 2.14: **Examples of Offset Local Attention Heads in all ViT Variants.** We find that all ViTs examined learn to use Offset Local Attention Heads, and some larger models even use ones with diagonal offsets. Midlines are drawn in red as a visual aid.

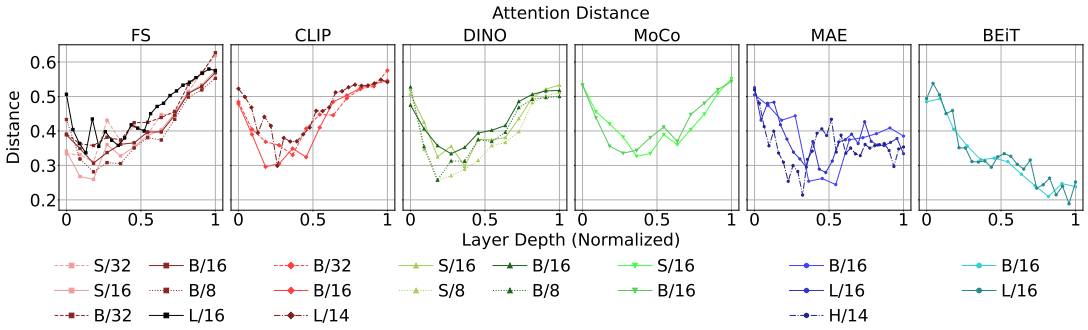


Figure 2.15: **Average Attention Distance for all ViT Variants.** Results are plotted against the normalized layer depth.

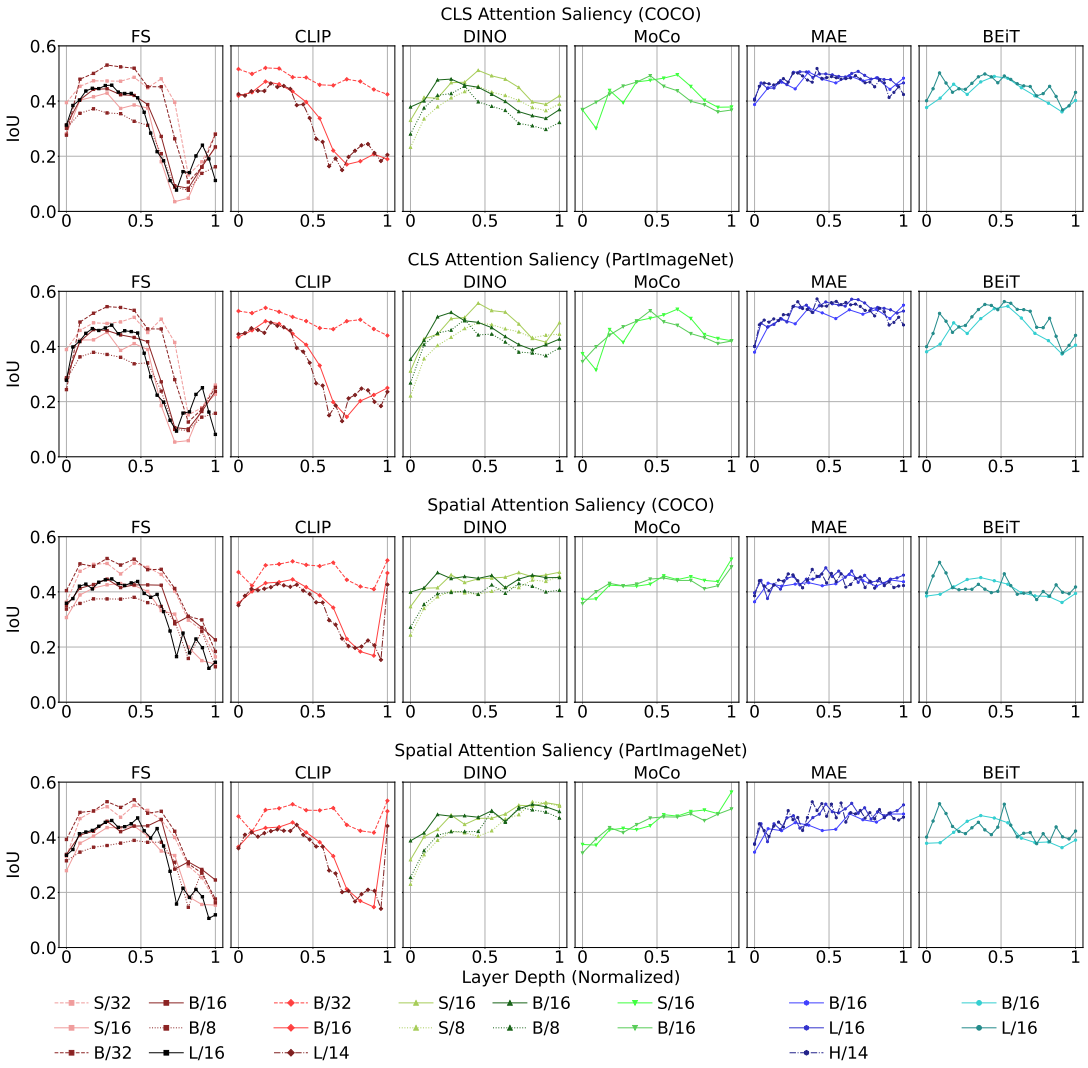


Figure 2.16: **Attention alignment with salient image content for all ViT Variants.** Results are shown for both CLS token attention and average spatial token attention on both COCO and PartImageNet. We see that the results are highly similar for the two datasets.

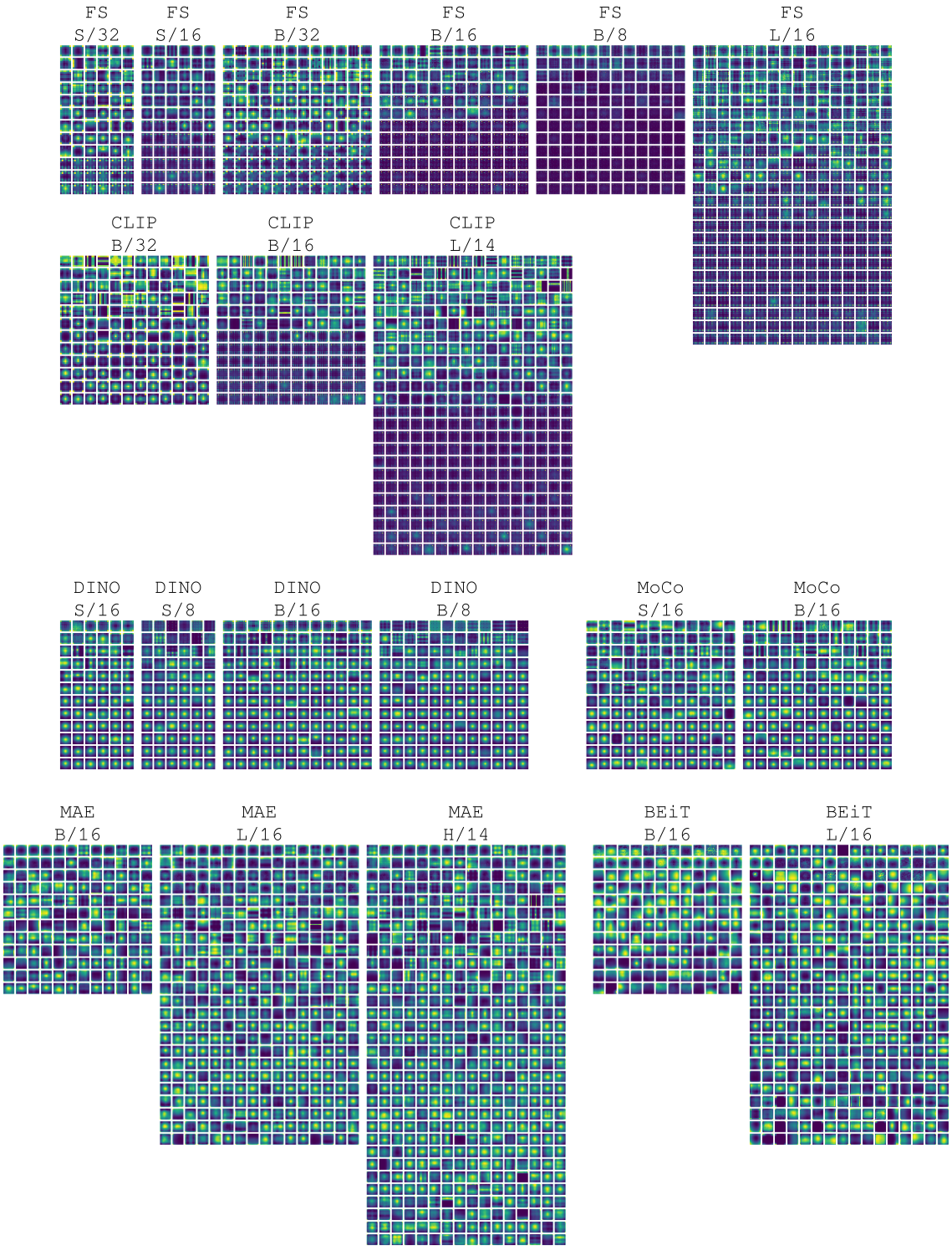


Figure 2.17: Average CLS token attention over 5000 images for every head of every ViT Variant.

Figure 2.18: Aligned Aggregated Attention Maps over 5000 images for every head of every ViT Variant.

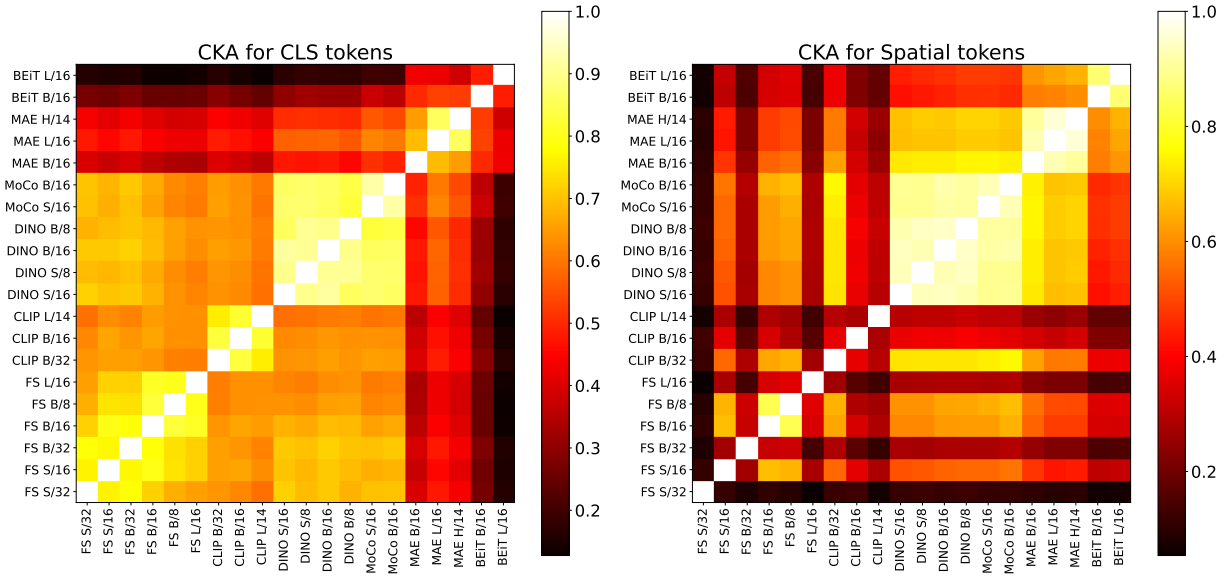


Figure 2.19: CKA similarity between final layer features of different ViTs for their CLS tokens (left) and spatial tokens (right).

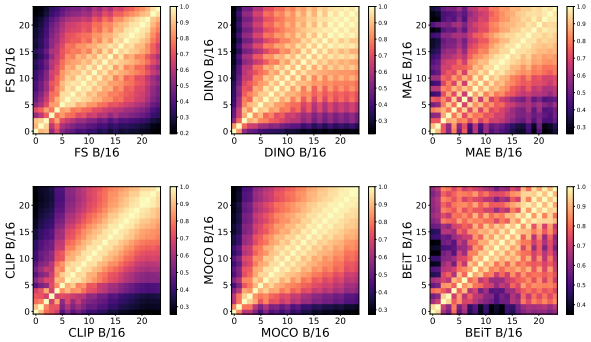


Figure 2.20: CKA for all ViT B/16 models showing their feature similarity across layers. Different supervision techniques result in different patterns.

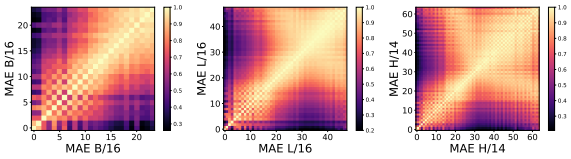


Figure 2.21: CKA across Base, Large and Huge MAE models.

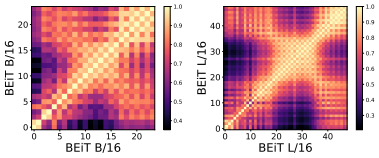


Figure 2.22: CKA across Base and Large BEiT models.

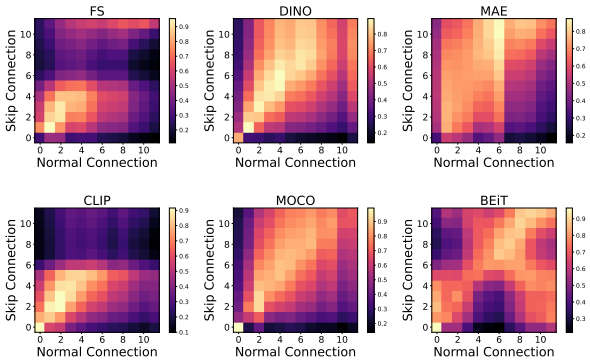


Figure 2.23: **Residual connection analysis.** We show the CKA similarity between the features coming from the skip-connect (Y-axis) and normal pathway (X-axis) for each MHA layer of each block. Each cell indicates the similarity between the skip connection features and output of normal pathway at that location.

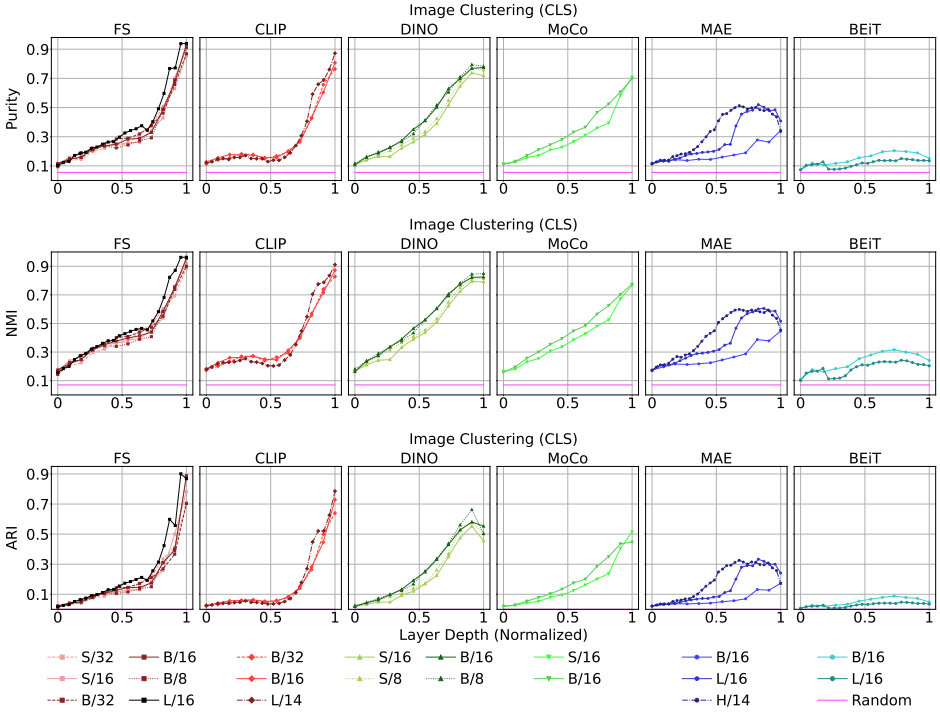


Figure 2.24: Expanded CLS feature clustering for image-level labels with ImageNet-50.

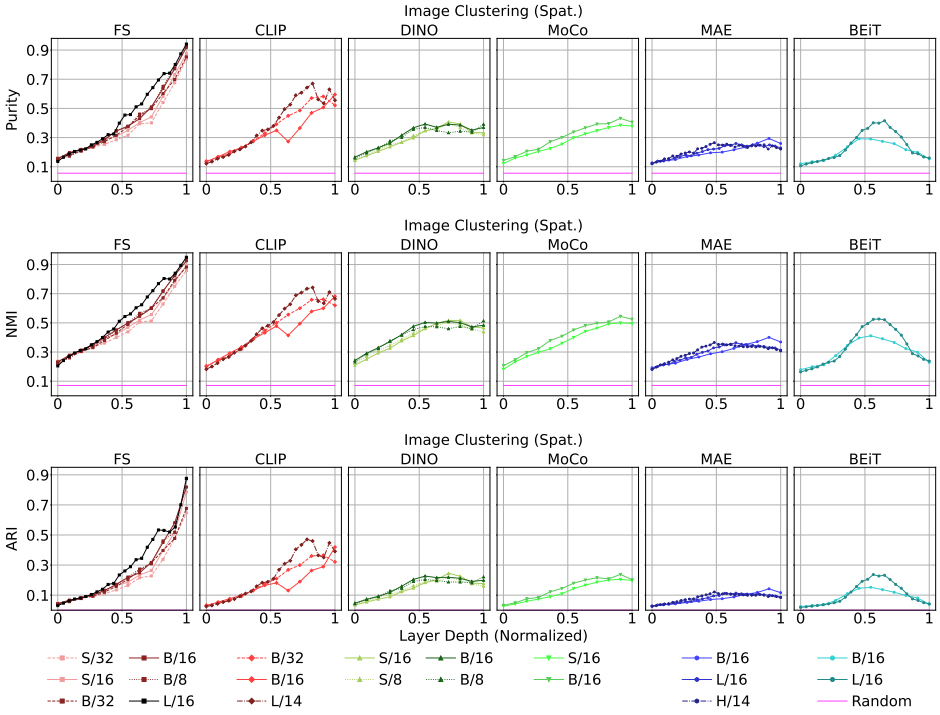


Figure 2.25: Averaged spatial feature clustering for image-level labels with ImageNet-50.

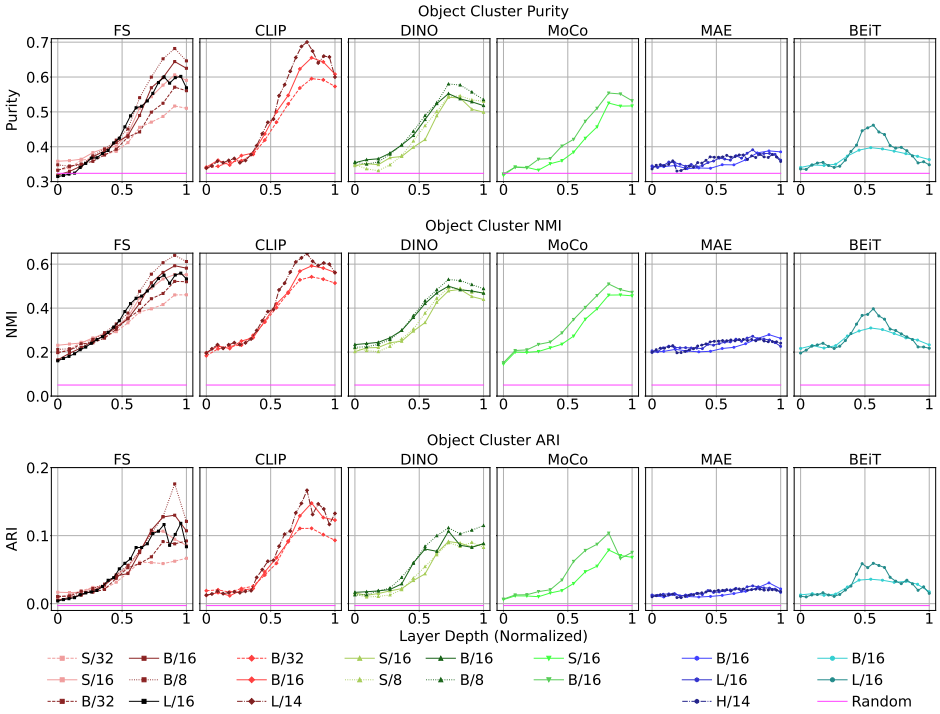


Figure 2.26: Expanded spatial feature clustering for object-level labels with COCO.

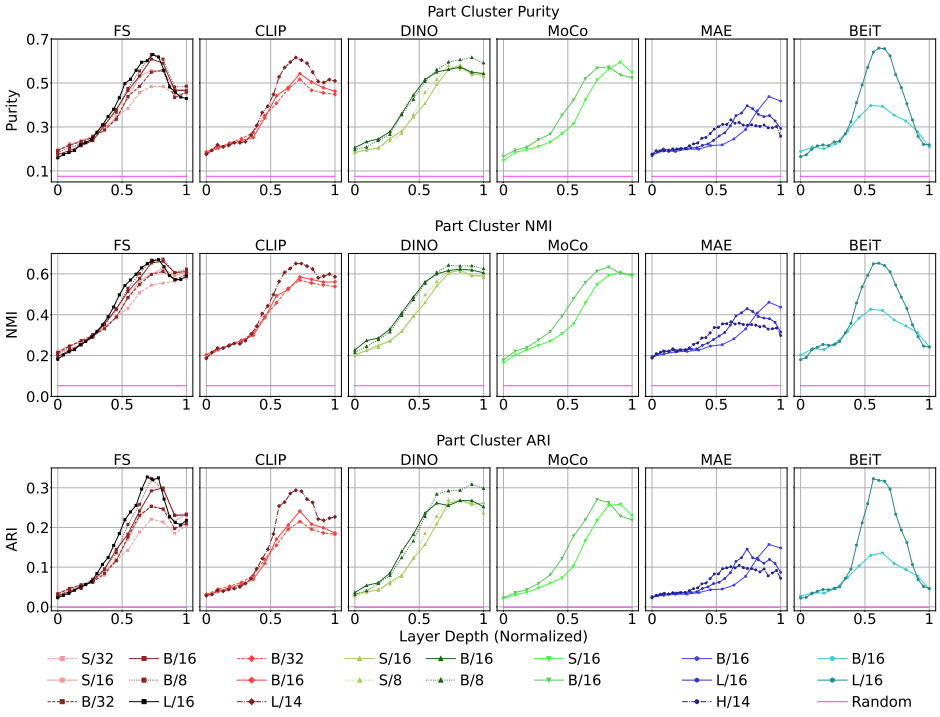


Figure 2.27: Expanded spatial feature clustering for part-level labels with PartImageNet.

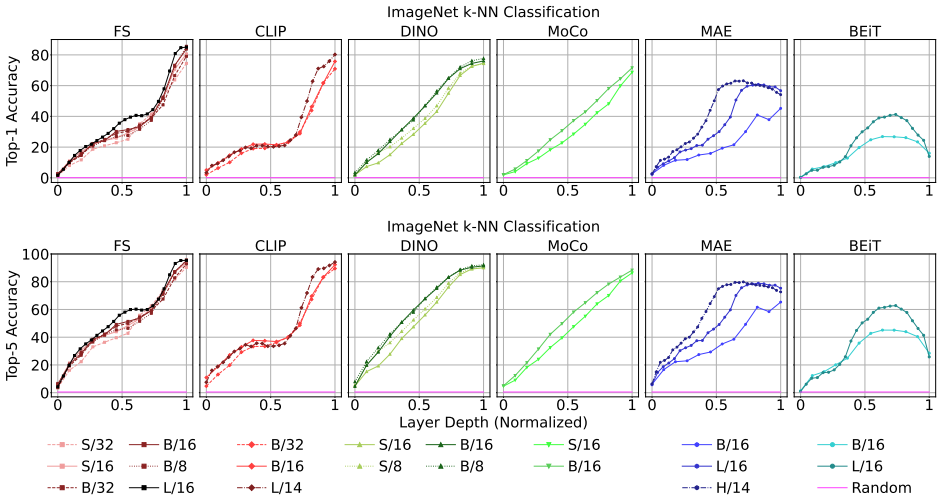


Figure 2.28: k-NN ImageNet classification results for all ViT variants.

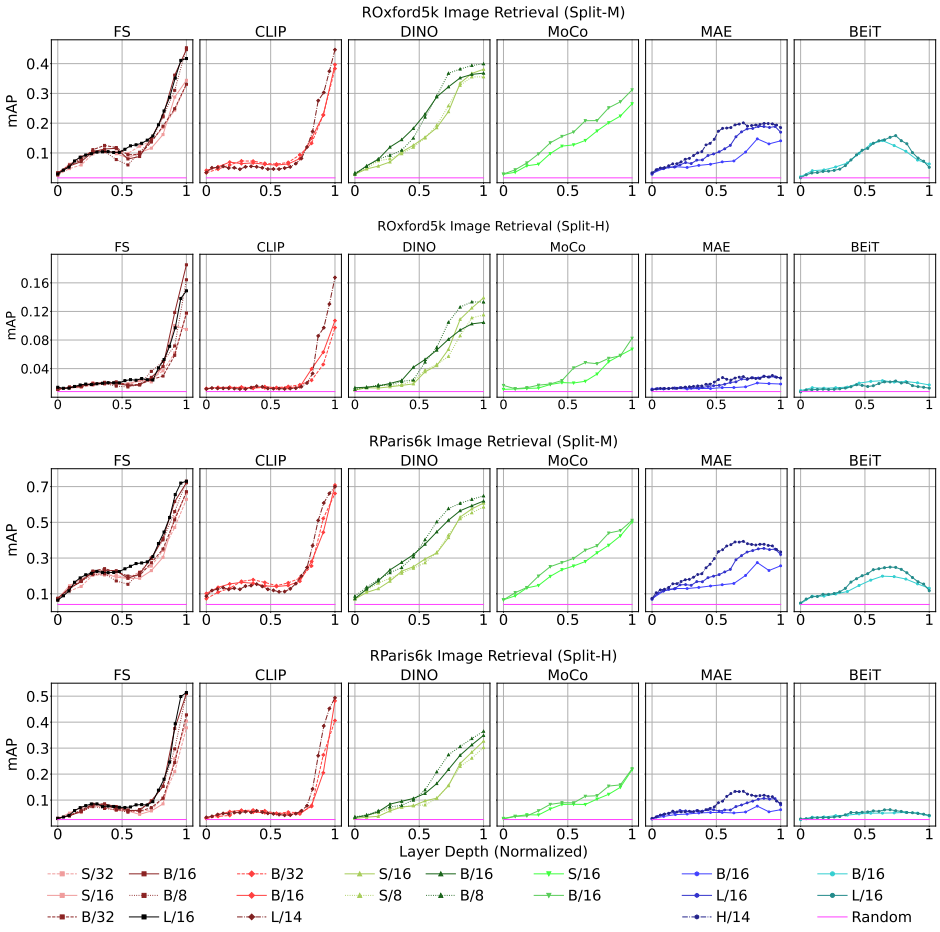


Figure 2.29: ROxford5k and RParis6k retrieval results for all ViT variants.

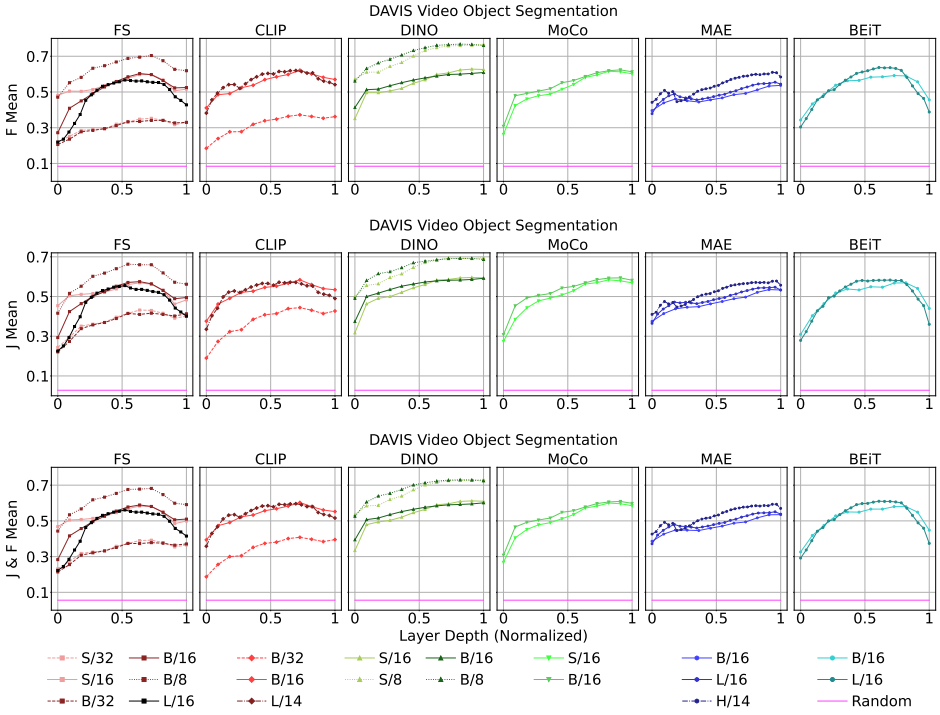


Figure 2.30: DAVIS Video Segmentation Propagation comparison for all ViTs

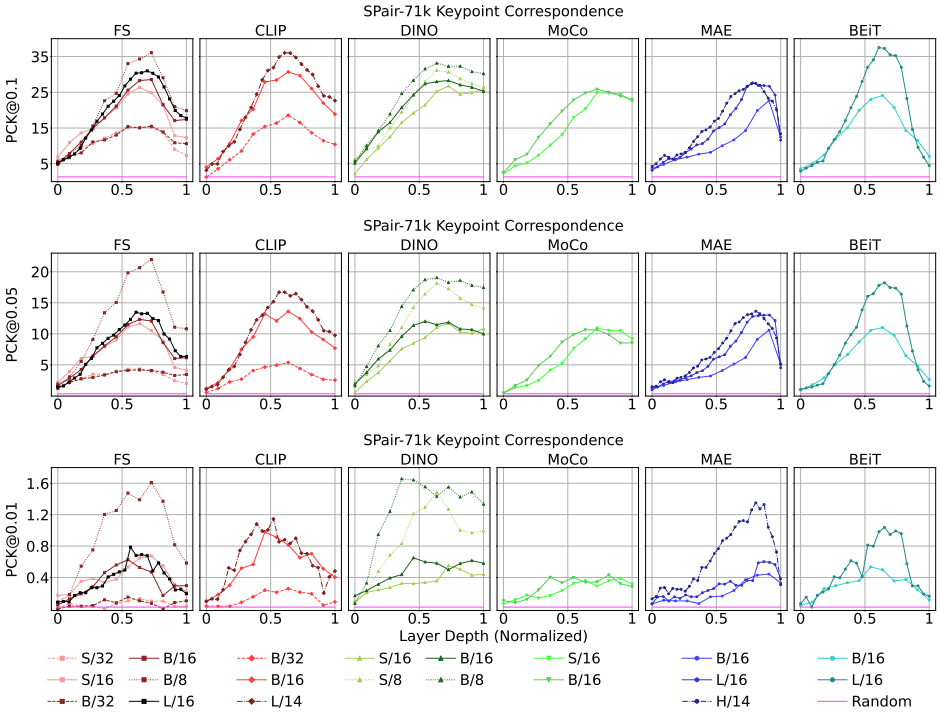


Figure 2.31: SPair-71k Keypoint Correspondence comparison for all ViTs

Chapter 3: SparseDet: Improving Sparsely Annotated Object Detection with Pseudo-positive Mining

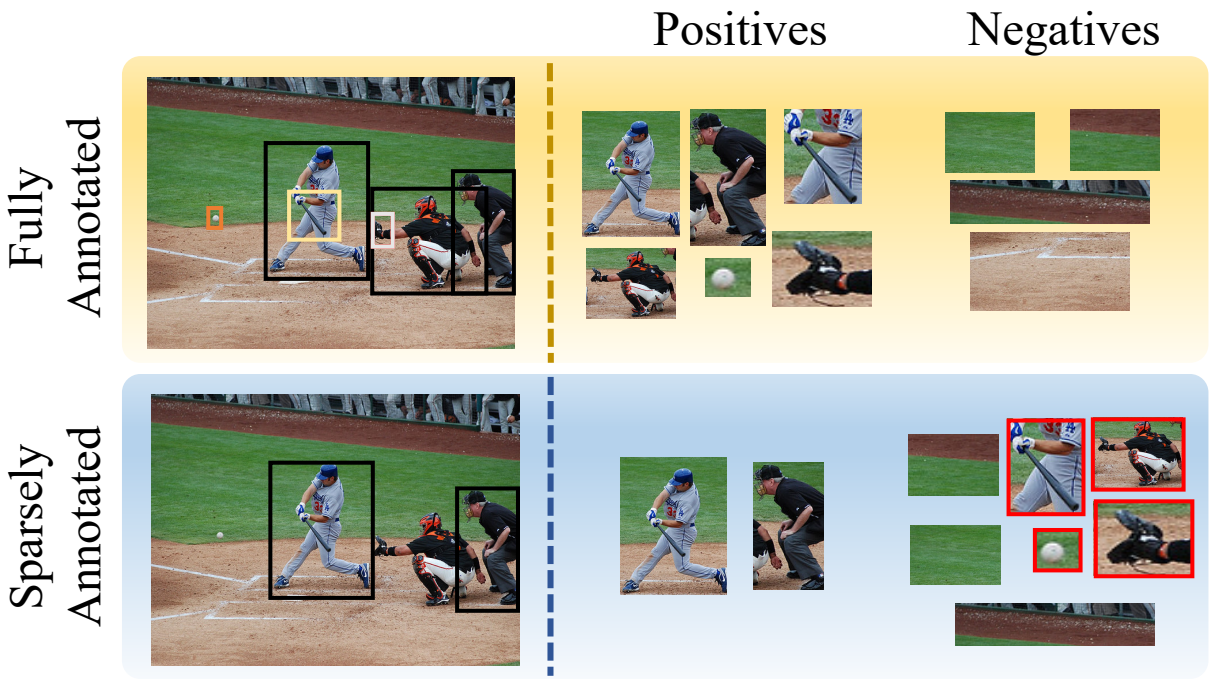


Figure 3.1: (Top) Most Object Detection datasets have exhaustive annotations for foreground/positives. During training, the unlabeled regions can be safely considered as background/negatives. Sparsely Annotated Object Detection datasets (bottom) have missing annotations. This results in foreground regions (shown in red) being considered as negatives during training, adversely affecting the performance of the classifier.

The performance of object detectors is sensitive to the quality of labeled data [90, 91, 92]. Existing object detection methods assume that the training data is pristine and a drop in performance is observed if this assumption fails. Noise in the data used for training object detectors can

arise due to incorrect class labels or incorrect/missing bounding boxes. In this work, we deal with the problem of training object detectors with sparse annotations, i.e., missing region or bounding boxes. This problem is of utmost importance, as obtaining crowd-sourced datasets [93, 94] can be expensive and laborious. The alternative is to use computer-assisted protocols to collect annotations which have been shown to be noisy and incomplete [95]. This problem of training object detectors with incomplete bounding box annotations is called Sparsely Annotated Object Detection (SAOD).

To understand why training with sparse annotations is detrimental to the performance, consider the example shown in Figure 3.1 (top). If the annotation were exhaustive, then the negative samples to the classifier contain *true* background regions. But with sparse annotations, as shown in Figure 3.1 (bottom), a few positive regions will inevitably be considered as negatives (shown in red), thereby wrongfully penalizing the classifier leading to lower performance. Existing methods[91, 96, 97, 98, 99] prevent this by predicting pseudo-labels and removing the foreground regions from the negatives to the classifier. However, at higher levels of sparsity, the quality of pseudo-labels is greatly affected, resulting in the same problem noted above.

Crowd sourced object detection datasets [1, 94, 100] are ensured to be almost exhaustively labeled. Hence, for SAOD, researchers artificially create sparsely annotated datasets from the original ones. There is no general consensus on the correct way to create the sparsely annotated datasets, *a.k.a.* splits, and hence each method reports results on one or two different splits. Split can be created by considering the dataset as a whole or per image (*i.e.*annotations can be removed by considering all the images or only a single image at once). They can also be created by removing annotations in a class-agnostic or class-aware fashion (*i.e.*remove $p\%$, of annotations per category or across all the categories). These variations result in splits with different data

distributions making some harder than the others (refer to Table 3.1). A proper benchmark that analyzes the performance of SAOD methods across these different types of splits is missing. This makes it difficult to compare methods and assess their effectiveness for a specific use case.

To tackle the issues discussed above, as our first major contribution, we present SparseDet, a novel SAOD framework that achieves state-of-the-art performance across multiple SAOD benchmarks in practice. SparseDet operates on an image and its augmented counterpart. The combination of features extracted from the two views is used to generate region proposals. Standard detection training methods, consider a region proposal as positive if its intersection over union (IoU) with any ground truth is greater than 0.5 and the rest are treated as negatives. This strategy works when the annotations are exhaustive, which implies that the remaining regions are from background. But due to missing annotations, some of these regions could belong to foreground instances. To prevent considering all region proposals without annotations as negatives, SparseDet partitions all the region proposals into labeled, *unlabeled* and background. The labeled and background regions are processed as usual. Features extracted from unlabeled regions are then trained with a self-supervised loss. Previous approaches like Co-Mining [99], consider two partitions, labeled and background, and generate pseudo-labels. This is a disadvantage at high sparsity as the generated pseudo-labels can be very noisy. The self-supervised loss in our approach enforces consistency between the features of the two views for the unlabeled regions and prevents penalizing the classifier due to false negatives.

Our second major contribution is unifying evaluation. The standard practice is to simulate sparsely annotated training data on COCO [94] and PASCAL-VOC [100] train sets and evaluate them on their corresponding standard validation set. As discussed above, a survey of recent SAOD approaches [91, 96, 97, 98, 99, 101, 102, 103, 104] reveals that there are at least ***five*** dif-

ferent ways to create splits, each differing in the strategy (refer to Section 3.3.2) used to achieve the desired level of sparsity. However, these splits have not been made public, making it difficult to replicate results for comparison. Additionally, each of these strategies has a different property for simulating sparse data, *e.g.*, different distribution of annotations per class, resulting in a different training set. As a result, methods trained on different splits cannot be compared with one another. To mitigate these issues we standardize the generation of these splits that enables the evaluation of any SAOD methods on all of them for fair comparison. Additionally, we propose a new benchmark that assesses the semi-supervised learning capabilities of SAOD methods, *i.e.*, leveraging unlabeled data to improve performance. We present our approach as a baseline. We will make the data for the new benchmark along with all the SAOD splits public to facilitate future research. We briefly summarize our contributions below.

- We propose a novel formulation, SparseDet, for SAOD which is an end-to-end approach that identifies labeled, unlabeled and background regions and deals with them in appropriate manner.
- We show state-of-the-art performance on sparsely annotated object detection across various splits. On average, we improve by 2.6, 3.9 and 9.6 mAP over previous state-of-the-art methods on three splits of increasing sparsity on COCO.
- We standardize the experimental setup for SAOD by evaluating methods on all the splits to facilitate future research. Additionally, we propose a new benchmark that evaluates the semi-supervised learning capabilities of SAOD methods.

In Section 3.1, we discuss the related works on SAOD and related fields. We describe our approach in detail in Section 3.2. We describe our experimental setup and present results in

Section 3.3, and conclude in Section 3.4.

3.1 Related Work

Semi-supervised object detection: Semi-supervised object detection (SSOD) is an active field that also deals with training object detectors with missing annotations. Existing works on semi-supervised object detection, have focused mainly on consistency regularization [105, 106] or pseudo-labeling-based approaches [107, 108, 109, 110]. The main idea behind these approaches is to perturb the images, or features, and apply a consistency regularization loss to enforce consistency between the predictions using a student teacher framework. However, typical SSOD methods assume a small exhaustively labeled set and a large unlabeled set for training. This is different from Sparsely annotated object detection (SAOD), which assumes a large training set which is sparsely labeled.

Sparsely annotated object detection: One of the initial works addressing this problem, by Niitani *et al*. [96], proposes utilizing logical relations between the co-occurrences of objects and pseudo-labeling. Yoon *et al*. [111] use object tracking to densely label objects across sparsely annotated frames along with single stage detectors to mitigate the negative effects of missing annotations. Wu *et al*. [91] propose a re-weighting approach where the gradients corresponding to region of interest are weighed as a function of overlap with ground truth instances. Improving upon the previous work, Zhang *et al*. [97] propose an automatic re-calibration strategy for single stage detectors where the negative branch is changed to take into account low confidence background predictions which might correspond to missing annotations.

Finally, one of the most recent works, Co-mining [99] uses a co-training strategy by using

two views of an image and predictions from one view along with the ground truth as supervision for the other view and vice versa. Our method doesn't solely rely on pseudo-labels and leverages a self-supervised loss to prevent propagating negative gradients to the model due to false negatives.

Fully supervised object detection: After the success of AlexNet [112] on the image classification challenge (ILSVRC 2012) [113], research on designing deep neural networks for object detection gathered more interest. First among the successful methods are the two-stage **Region-based convolutional networks** (R-CNN) family of detectors. Two-stage object detectors consists of 1) a region proposal stage which produces a set of candidate object bounding boxes followed by 2) a classification stage which classifies each candidate region as either belonging to a foreground object or "background". R-CNN processes a large amount of region proposals by cropping the input image and using a CNN backbone to extract features making it extremely slow. **Fast R-CNN** [114] was proposed to overcome this limitation. Fast R-CNN computes one convolution feature map for the whole image. RoI Pooling was introduced to pool the feature for each region of interest into a fixed spatial dimension. RoI pooling shares the computation among all the region proposals speeding up training and inference. Fully connected layers are applied on the fixed RoI pooled feature maps which are then passed to two sister heads, for classification and bounding box regression. The whole network is trained end-to-end with a multi-task loss avoiding the multi-stage training in R-CNN [115]. While Fast R-CNN improved the efficiency of its predecessor, test time computation bottleneck is still an issue because of the region proposal method employed. **Faster R-CNN** [116] proposed a region proposal network (RPN), an elegant solution that trains deep networks to predict region proposals for practically no additional computational overhead. RPN shares convolutional layers with Fast R-CNN [114] and at test time the cost of generating proposals is minimal. Faster R-CNN paved the way for more

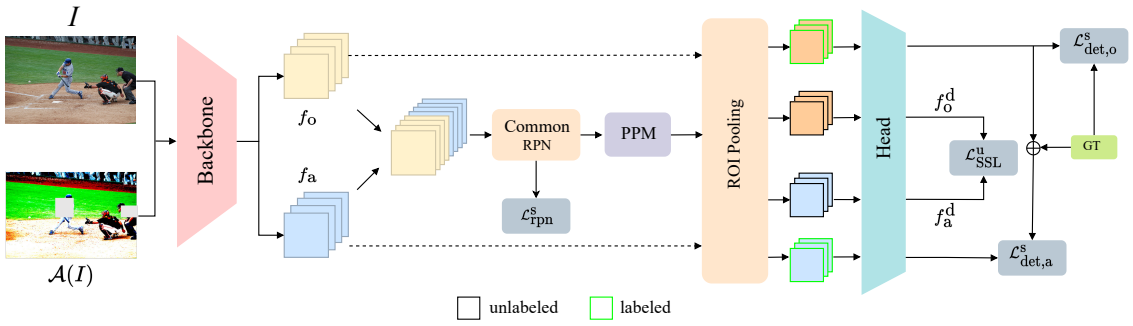


Figure 3.2: Illustration of SparseDet for sparsely annotated object detection. Following feature extraction from the original and augmented image, a common set of proposals is generated by the common RPN (C-RPN). Using an end to end approach, we identify and mine proposals corresponding to missing annotations using pseudo positive mining (PPM). We train the network end-to-end using a combination of supervised and self-supervised losses. The unlabeled instances (black) are supervised with a self-supervised loss and the labeled instances (green) are supervised with ground truth annotations.

sophisticated and efficient two stage detection architectures [117, 118, 119, 120, 121]. Faster R-CNN has also been extended to achieve state-of-the-art instance segmentation [122], panoptic segmentation [123, 124], and 3D mesh generation [125] *etc*.. A few limitations of two stage object detectors and anchor boxes has also inspired the family of single-stage [126, 127, 128] and anchor-free object detection systems [129, 130].

3.2 Approach

We present SparseDet for Sparsely Annotated Object Detection. Given N images, we denote the set of labeled regions in the dataset as $\mathcal{B}_{\text{l}} = \{b_i, c_i\}_{i=1}^{\text{N}_\text{l}}$ where (b_*, c_*) are the bounding box and class labels respectively. The unlabeled regions are denoted as $\mathcal{B}_{\text{ul}} = \{b_k\}_{k=1}^{\text{N}_\text{ul}}$ and the background regions as $\mathcal{B}_{\text{bg}} = \{b_q\}_{q=1}^{\text{N}_\text{bg}}$. Note that unlike existing SAOD approaches, we do not assume images to contain at least one labeled region in this work. The unlabeled $\mathcal{B}_{\text{ul}}$ and background $\mathcal{B}_{\text{bg}}$ sets are not known a-priori.

3.2.1 Overview

The proposed approach is shown in Figure 3.2 and consists of a backbone network that extracts features from the original and augmented views of an image. The common RPN (C-RPN), concatenates the features, to generate a set of region proposals. A region proposal b can belong to one of three groups, namely 1) labeled regions $b \in \mathcal{B}_{\text{l}}$, 2) unlabeled foreground regions $b \in \mathcal{B}_{\text{ul}}$, or 3) background regions $b \in \mathcal{B}_{\text{bg}}$. For a given set of ground-truth annotations, the first group, i.e. labeled regions can be automatically identified. The problem then reduces to identifying and separating the second group i.e. the unlabeled regions, from the background regions. Given all the region proposals, a pseudo-positive mining (PPM) step identifies the unlabeled regions and segregates them from the background regions. The labeled and unlabeled regions are trained using supervised and self-supervised losses respectively. We describe each stage in detail below.

3.2.2 Feature Extraction

Given an image I, an augmented version of I denoted as $\mathcal{A}(I)$ is computed. In this work, we use random contrast, brightness, saturation, lighting and bounding box erase in a cascaded fashion to generate $\mathcal{A}(I)$. A backbone network is employed to extract two features, f_{o} and f_{a}, from I and $\mathcal{A}(I)$ respectively.

3.2.3 Common RPN (C-RPN)

Two stage object detectors [116, 118, 131] use a region proposal network (RPN) [116] to generate regions of interest (RoIs). We propose C-RPN which concatenates f_{o} and f_{a} to obtain the RoIs. This is different from existing approaches that use f_{o} and f_{a} to generate two separate

sets of RoIs. Operating on two sets increases the difficulty of identifying the labeled, unlabeled and background regions which are required for subsequent stages. The sparse ground truth is used as supervision to train the C-RPN with a binary cross entropy loss for foreground classification and smooth L1 [132] loss for bounding box regression as shown below:

$$\mathcal{L}^{\text{s}}_{\text{rpn}}\left(\mathbf{c}, \mathbf{b}, \mathbf{c}^*, \mathbf{b}^*\right) = \mathcal{L}_{\text{bce}}\left(\mathbf{c}, \mathbf{c}^*\right) + \lambda \mathcal{L}_{\text{reg}}\left(\mathbf{b}, \mathbf{b}^*\right) \tag{3.1}$$

where $\mathbf{c}$, $\mathbf{b}$ are the objectness scores and bounding boxes, and $\mathbf{c}^*$, $\mathbf{b}^*$ are the corresponding ground truth.

3.2.4 Pseudo Positive Mining (PPM)

Given the RoIs from C-RPN, the next step is to seggregate the unlabeled regions from the labeled and background regions. A standard practice in training detectors is to consider all proposals with an objectness score greater than τ_{obj} and IoU greater τ_{fg} with any ground truth as foreground and proposals with IoU less than τ_{bg} as background. In the presence of missing annotations, detectors are wrongly penalized because the IoU of RoIs corresponding to missing annotations with any ground truth is less than τ_{bg}. To avoid this, we mine "potential" foregrounds and consider them as unlabeled regions ($\mathcal{B}_{\text{u}}$). The proposed PPM module is based on our observation that when trained with sparse annotations, the RPN can reliably distinguish foreground from background regions. We pick all RoIs that have an objectness score greater than τ_{ppm} and IoU less than τ_{bg} with any ground truth as the unlabeled regions. The remaining RoIs are assigned as background.

3.2.5 Losses

The pseudo positive mining step segregates the RoIs into labeled, unlabeled and background regions. An RoI pooling layer [116] extracts RoI pooled features for the labeled and background regions using the feature f_{o}. The detection head processes the RoI pooled features to predict class-wise probabilities and bounding box adjustments for each region. The ground truth is used to supervise the predictions by applying the cross entropy loss for classification and smooth L1 [132] for bounding box regression as shown below:

$$\mathcal{L}^{\mathrm{s}}_{\mathrm{det,o}}\left(\mathbf{c}, \mathbf{b}, \mathbf{c}^{*}, \mathbf{b}^{*}\right) = \mathcal{L}_{\mathrm{ce}}\left(\mathbf{c}, \mathbf{c}^{*}\right) + \lambda \mathcal{L}_{\mathrm{reg}}\left(\mathbf{b}, \mathbf{b}^{*}\right) \tag{3.2}$$

where $\mathbf{c}$, $\mathbf{b}$ are the class and bounding box predictions, and $\mathbf{c}^{*}$, $\mathbf{b}^{*}$ are the corresponding ground truth class labels and bounding boxes.

All detections with a score greater than τ_{m} are combined with the ground truth followed by a class specific NMS to obtain the merged ground truth. It is ensured that no ground truth annotation is discarded during this step. The labeled and background regions along with f_{a} are used to extract RoI pooled features which are then fed to the detector head. The predictions of the detector head are supervised with the merged ground truth with the following losses:

$$\mathcal{L}^{\mathrm{s}}_{\mathrm{det,a}}\left(\mathbf{c}, \mathbf{b}, \mathbf{c}^{*}, \mathbf{b}^{*}\right) = \mathcal{L}_{\mathrm{bce}}\left(\mathbf{c}, \mathbf{c}^{*}_{\mathbf{m}}\right) + \lambda \mathcal{L}_{\mathrm{reg}}\left(\mathbf{b}, \mathbf{b}^{*}_{\mathbf{m}}\right) \tag{3.3}$$

where $\mathbf{c}$, $\mathbf{b}$ are the class and bounding box predictions from the detector head, and, $\mathbf{c}^{*}_{\mathbf{m}}$, $\mathbf{b}^{*}_{\mathbf{m}}$ are the corresponding merged ground truths.

Finally, a class agnostic NMS is performed on the unlabeled regions $\mathcal{B}_{\text{u}}$ (obtained after PPM described in Sec. 3.2.4). The unlabeled regions along with f_{o} and f_{a} are passed through the ROI pooling layer and the detection head to obtain f_{o}^{d} and f_{a}^{d} respectively. A self-supervised loss is applied that enforces the detection head features of the original and augmented regions to be consistent with each other as shown below:

$$\mathcal{L}_{\text{SSL}}^{\text{u}}\left(f_{\text{a}}^{\text{d}}, f_{\text{o}}^{\text{d}}\right) = \|f_{\text{a}}^{\text{d}} - f_{\text{o}}^{\text{d}}\|_2^2 \tag{3.4}$$

The network is trained end-to-end with both supervised and unsupervised losses as shown below:

$$\mathcal{L} = 0.5 * \left(\mathcal{L}_{\text{det,o}}^{\text{s}} + \mathcal{L}_{\text{det,a}}^{\text{s}}\right) + \mathcal{L}_{\text{rpn}}^{\text{s}} + \mathcal{L}_{\text{SSL}}^{\text{u}} \tag{3.5}$$

Discussion: In the absence of ground truth annotations, i.e. for completely unlabeled images, the supervised losses ($\mathcal{L}_{\text{det,o}}^{\text{s}}$, $\mathcal{L}_{\text{det,a}}^{\text{s}}$, $\mathcal{L}_{\text{rpn}}^{\text{s}}$) cannot be computed. Our proposed approach leverages self-supervised consistency loss that does not need ground truth. This helps our approach leverage these unlabeled regions unlike contemporary SAOD methods. We claim that this is the first method to use self-supervised losses for SAOD. Even though Co-Mining [99] claims to use self-supervised learning it is technically co-learning. Ours is the first approach to use pseudo-labeling and a self-supervised loss to handle the sparse annotations.

Table 3.1: **Sparsely annotated object detection** results on three splits of COCO dataset. "Oracle" corresponds to training models using all annotations. Results are reported on the COCO validation set using AP[0.50:0.95].

Method	Split-1			Split-2			Split-3			100%
	30%	50%	70%	30%	50%	70%	30%	50%	70%	
Oracle										40.91
Pseudo Label [96]	-	27.50	-	-	-	-	-	-	-	-
BRL [97]	-	32.70	-	-	-	-	-	-	-	-
Co-mining [99]	36.35	32.84	24.93	36.72	33.04	24.83	36.76	32.54	24.96	-
Ours	**38.22**	**35.92**	**32.68**	**39.76**	**36.94**	**35.33**	**39.56**	**37.15**	**35.48**	-

Table 3.2: **Sparsely annotated object detection** results on two splits of PASCAL-VOC dataset. "Oracle" corresponds to training models using all annotations. Results are reported on VOC 07 test set usiing AP_{50}.

Method	Split-4			Split-5		
	Easy	Hard	Extreme	30%	40%	50%
Oracle		83.09			77.47	
BRL [97]	73.50	71.70	66.20	-	-	-
Co-mining [99]	79.59	78.38	69.60	74.42	73.30	69.89
Ours	**82.15**	**81.50**	**75.59**	**76.84**	**75.88**	**74.35**

Table 3.3: Results on the proposed **SSL+SAOD** setup. VOC12 [1] is used as the unlabeled data and p is the removal percentage

Method	AP		
	30%	40%	50%
Co-mining [99]	46.53	45.98	43.21
Ours	**48.34**	**47.82**	**46.76**

3.3 Experimental Evaluation

In this section, we describe the experiments to evaluate our proposed approach. In Sections 3.3.1 and 3.3.2, we describe data, splits, and metrics. In Section 3.3.3, we mention the implementation details followed by the baselines in Section 3.3.4. We compare with contemporary methods in Section 3.3.5, followed by an ablation study in Section 3.3.6.

3.3.1 Data and Metrics

We conduct all our experiments on the COCO [94] and PASCAL-VOC [1, 100] (2007+2012) datasets. The COCO [94] dataset consists of 118000 and 5000 images for training and validation respectively. Experiments on the PASCAL-VOC07 [100] are conducted on 5011 trainval images

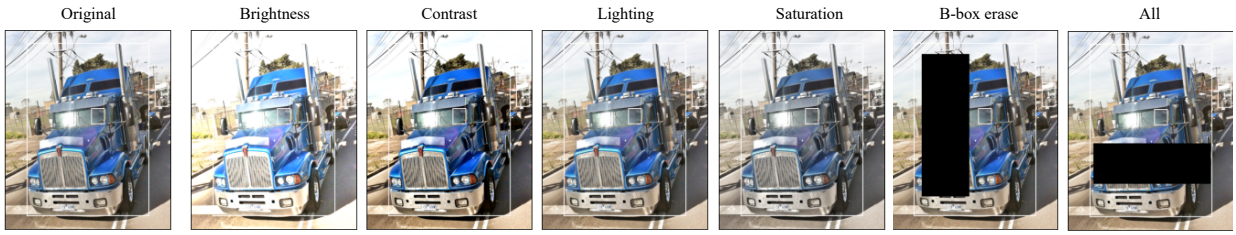


Figure 3.3: Illustration of the effects of various augmentations used in this work.

and performance is computed on 4952 images of the test set. The PASCAL-VOC12 version consists of 11530 (trainval) images for training and evaluation is done on the PASCAL-VOC07 test set. Following past literature [91, 96, 97, 98, 99], we create five different splits (Section 3.3.2) and report results on them. For splits on the COCO [94] dataset, we use the standard COCO style Average Precision (AP). For splits on the PASCAL-VOC [100] dataset, we use the standard PASCAL-VOC style AP_{50}, which is Average Precision computed at an IoU threshold of 0.5.

3.3.2 SAOD Splits

An extensive review of sparsely annotated object detection methods reveals that there are atleast five popular types of splits in use for creating training data. Most methods report results on a subset of these making it harder to compare across methods.

We standardize the evaluation of SAOD methods by evaluating exhaustively on all the splits facilitating future research. We briefly describe the splits below.

Split-1 [98, 99]**:** In this split, for each object category, $p\%$ of annotations are randomly removed from the COCO [94] train set and results are reported on the validation set. This split simulates sparsity at dataset level in a class aware fashion. Note, this split can contains images with no annotations. We experiment with $p = \{30, 50, 70\}$.

Split-2: For each image in the COCO [94] train set, all annotations of $p\%$ of all categories in that

image removed and results are reported on the COCO validation set. This split can be considered as image level and class aware. We experiment with $p = \{30, 50, 70\}$. Note, this split might contain images with no annotations.

Split-3: This split, which uses the COCO 2017 [94] train set for training and the validation set for evaluation, deletes $p\%$ annotations in a class agnostic fashion for each image ensuring at least one annotation. This split is image level but class agnostic. For the experiments, we use $p = \{30, 50, 70\}$.

Split-4 [97, 99]**:** This split requires evaluating models on three different settings namely *easy*, *hard* and *extreme* ensuring at least one annotation per image constructed using PASCAL-VOC 2007+12 [1, 100] trainval set. Results are reported on the PASCAL-VOC 2007 [100] test set. For each image in the training set, the easy setting randomly removes one annotation while the *hard* setting randomly removes half of the annotations. The *extreme* setting retains only one annotation per image. All the sets ensure each image consists of atleast one annotation. This split is a small scale version for an image level class agnostic split.

Split-5 [91]**:** This split uses the PASCAL-VOC 2007 [100] train set for training and the PASCAL-VOC 2007 test set for evaluation and drops $p\%$ annotations per class. For each image in this construction, instances of randomly selected categories are exhaustively annotated while the remaining categories do not have any annotations. This is a small scale version of Split-1 as annotations are dropped in a class aware manner across the full dataset maintaining atleast 1 annotation per image. In this case we use $p = \{30, 40, 50\}$.

3.3.3 Implementation details

For all our experiments, we use a Faster RCNN [116] architecture with a ResNet-101 [133] FPN backbone [118] implemented using Detectron2 [134] framework. For the image augmentations, we apply a For augmentation, a cascade of random contrast, brightness, saturation, lighting and bounding box erase augmentations. The effect of this augmentation is shown in Figure 3.3. We train all our models with a batch size of 8 for 270000 and 18000 iterations on the COCO and PASCAL-VOC splits respectively with a learning rate of 0.01. The learning rate is decreased ten fold twice at $\{210000, 250000\}$ and $\{12000, 15000\}$ for COCO and PASCAL-VOC respectively. We adopt a warm-up strategy for 1000 and 100 iterations for the COCO and PASCAL-VOC respectively. Following existing detector implementations we set $\tau_{\text{bg}}, \tau_{\text{fg}}$, τ_{obj} and τ_{ppm} to $0.2, 0.4$, 0.5 and 0.8 respectively. We set τ_m to 0.9. During inference, we compute the backbone features (f_{o} and f_{a}) for both the original and augmented versions of the input to obtain the RoIs and only f_{o} is used for the final detections.

3.3.4 Baselines

We compare our method against Co-Mining [99], BRL [97], and Pseudo Label [96].

We choose these methods as they outperform previous approaches for the SAOD task. We use the public implementation of these methods and usa a ResNet 101 FPN [118] backbone for all the experiments in order to perform a fair comparison.

Table 3.4: Ablation of various components of proposed approach.

C-RPN	PPM	$\mathcal{L}^{u}_{SSL}$	$\mathcal{L}^{s}_{det,a}$	AP	Δ
✗	✗	✗	✗	41.77	–
✗	✗	✗	✓	42.67	0.90
✓	✗	✗	✓	45.30	3.53
✓	✓	✗	✓	45.44	3.67
✓	✓	✓	✗	45.51	3.74
✗	✓	✓	✓	44.86	3.09
✓	✓	✓	✓	**46.00**	4.23

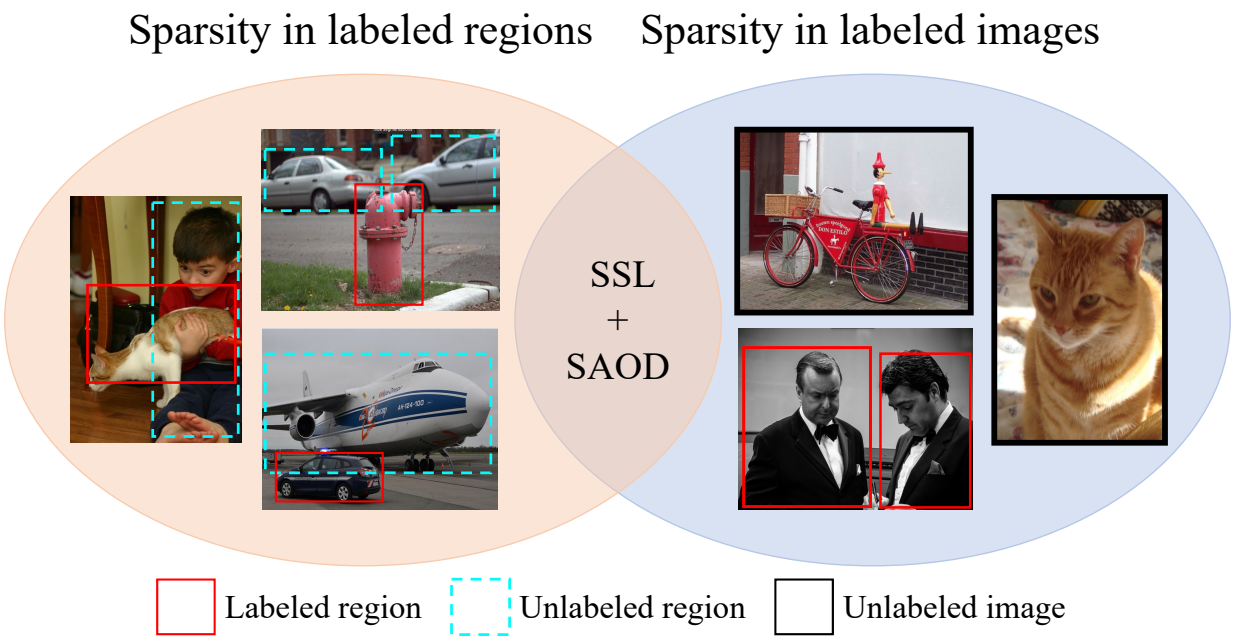


Figure 3.4: Types of sparsity in labeled data; sparsity in labeled regions (left) and images (right). Our proposed SSL+SAOD is a realistic setup that presents challenges from both kinds of sparsity.

3.3.5 Comparison with state-of-the-art

In this section, we compare our method with contemporary methods. We evaluate all the models in two different setups. We name the first setup *Sparsely annotated* setup which evaluates models on SAOD. The second setup, contains labeled and unlabeled images and regions.

***Sparsely annotated* setup:** We show results of this setup in Table 3.1 and Table 3.2. Splits-2 and 3 ensure at least one annotation per image is retained. On the other hand, Split-1 doesn't ensure this and has significantly more unlabeled data than the other splits. Splits-4 and 5 have no unlabeled images at all. In both the tables, the rows named "oracle" refers to the models trained using all annotations.

From Table 3.1, we observe that our approach outperforms the other baselines and is closer to the oracle performance on the 30% splits. Co-mining [99], performs competitively on all the splits at 30%.

On Split-1, our method obtains a performance improvement of 1.87, 3.08 and 7.75 on 30%, 40% and 50% sparsity respectively over Co-mining [99].

On Split-2, our method obtains an improvement of 3.04, 3.9 and 10.5 at 30%, 40% and 50% sparsity respectively over the previous state of the art Co-mining. On Split-3, an again we see consistent improvement of 2.8, 4.61 and 10.52 at 30%, 40% and 50% sparsity respectively over Co-mining.

In particular, on the hardest setting (70%) in Split 1-3, we demonstrate that current SAOD methods struggle at higher sparsity in the labeled data. This is because the performance of current SAOD approaches rely on the quality of pseudo labels which degrades at higher sparsity. Our proposed PPM prevents penalizing the classifier irrespective of the quality of pseudo labels and utilizes self-supervised loss to benefit from mined regions. On average, all the methods achieve lower performance on Split-1 compared to the other splits due to the nature of its creation, *i.e.*, class aware dataset level fashion of sparsity.

From Table 3.2, we see a similar trend as above. All the methods perform competitively on

the easier settings of both splits. However, the performance gap between the our approach and Co-mining [99] increases at higher sparsity.

On Split-4, we get an improvement of 2.56, 3.12 and 5.99 (AP_{50}) percentage points on the *easy*, *hard* and *extreme* settings respectively over Co-mining. On Split-5, an improvement of 2.42, 2.58 and 4.46 points was observed.

***Single Instance Object Detection (SIOD)* setup**: [135] proposed SIOD where only one instance per category is annotated in every image. While this setup has its benefits [135], note that it is a special case of SAOD. For this setup, we obtain an AP of 32.89 (compared to 31.9 of [135]) which is an improvement of $\sim$1 mAP.

Table 3.5: Analysis of the threshold used for PPM.

Threshold	0.6	0.7	0.8	0.9	0.95
AP	45.96	45.96	**46.00**	45.77	45.30

Table 3.6: Removing test time augmentations.

Method	**Split-1**			**Split-2**			**Split-3**		
	30%	50%	70%	30%	50%	70%	30%	50%	70%
TTA	38.22	35.92	32.68	39.76	36.94	35.33	39.56	37.15	35.48
w/o TTA	38.32	35.91	32.67	39.77	36.95	35.27	39.55	37.12	35.48

***SSL+SAOD* setup:** We propose a semi-supervised learning benchmark for SAOD. This benchmark entails training models on a sparsely annotated labeled and an unlabeled set. As shown in Figure 3.4, this setup introduces two kinds of sparsity in the data, namely, sparsity in labeled

regions (left) and images (right). We believe this is a realistic setup and SAOD methods must be capable of tackling both these kinds of sparsity in the data. For this setup, we use Split-5 (Section 3.3.2) with increasing sparsity as the labeled set and VOC12 trainval as the unlabeled set. We use the COCO-style AP metric to report results on this setup.

In Table 3.3, we compare against Co-mining [99] and observe an improvement of 1.81, 1.84 and 3.55 mAP on the 30%, 40% and 50% sparsity levels respectively. We observe that the gap in performance increases with sparsity, consistent with our observation for Tables 3.1 and 3.2. This can be attributed to the inability of methods like Co-mining to handle unlabeled images and high sparsity in the labeled data. We will make this benchmark public and propose this method as a baseline. We encourage researchers to report results on this benchmark in the future.

3.3.6 Ablation Experiments

In this section we conduct ablation experiments to understand the various components. For the ablation experiments, we use a ResNet-101 as the backbone network and train on Split-5 with $p = 50\%$. We evaluate on the VOC07 test set and report the COCO style AP metric.

From Table 3.4, a baseline model trained on the ablation set attains 41.77 (row 1). Pseudo-labeling $\left(\mathcal{L}^{s}_{\text{det},a}\right)$ improves the performance by 0.9 (row 2). Co-mining [99] relies extensively on pseudo-labels. This results in a drop in performance at higher sparsities due to noisy pseudo-labels. Addition of the C-RPN improves our performance by 3.53 points (row 3). C-RPN reduces the overhead of computing two sets of proposals and learns a better notion of objects due to the combined processing of features from the two views.

The combination of C-RPN and PPM improves the performance by 3.67 (row 4). We do

Table 3.7: Comparison with a different backbone. Results are reported on the COCO validation set on Splits 1-3 using AP and on VOC 2007 test set on Splits-4,5 using AP_{50}.

Method	**Split-1**			**Split-2**			**Split-3**			**Split-4**			**Split-5**		
	30%	50%	70%	30%	50%	70%	30%	50%	70%	Easy	Hard	Extreme	30%	40%	50%
Ours (C4)	37.67	35.95	33.16	39.22	36.81	34.98	38.84	36.76	35.26	80.56	80.43	74.45	77.38	76.13	75.32
Ours (FPN)	38.22	35.92	32.68	39.76	36.94	35.33	39.56	37.15	35.48	82.15	81.50	75.59	76.84	75.88	74.35

not observe major improvements with the introduction of PPM because its task is to identify and segregate the unlabeled regions from the backgrounds. After the segregation, PPM does no further processing to improve performance. The power of PPM can be observed when trained in conjunction with consistency regularization loss ($\mathcal{L}^{u}_{SSL}$) which achieves the best performance of 46 (row 7); an improvement of 4.23 points over the baseline. We distinguish ourselves from pseudo-labeling approaches like Co-mining in one important aspect. While, Co-mining [99] relies solely on pseudo-labels, we leverage additional components from self-supervised learning along with pseudo-labeling. In row 5, due to C-RPN, PPM and $\mathcal{L}^{u}_{SSL}$, we show an improvement of 3.74 over the baseline. Our proposed components are orthogonal to pseudo-labeling as using them together results in an additional improvement of ~ 0.5 on mAP. At higher sparsity in the labeled dataset, the proposed C-RPN, PPM and $\mathcal{L}^{u}_{SSL}$ are less affected than pseudo-labels resulting in the large improvements on these splits. Finally, we show the effect of C-RPN by generating a single set of proposals from the original image and using it for both the branches. We observe a drop in performance (row 6) highlighting the effectiveness of C-RPN.

PPM mines potential positives which can be mistaken for negatives due to missing annotations. We rely on the objectness score of the RPN to identify these regions. In Table 3.5, we vary the threshold of PPM. For a low threshold, a few hard negatives might also be identified as pseudo-positive leading to a drop in performance. With a high threshold, a few potential positives

might not be mined. We observe that a threshold of 0.8 provides a good trade-off and is therefore used for all experiments unless stated otherwise. In all our experiments, PPM is performed after an initial warmup of 9000 and 30000 iterations on the PASCAL-VOC and COCO datasets respectively.

3.3.6.1 Inference without augmentations

Our method requires passing an input with augmentations during inference as well. It should be noted that this is not a test-time augmentation (TTA), a technique that typically involves passing images at a higher resolution. We perform inference by removing the augmentation and extracting the region proposals using C-RPN. We show the results in Table 3.6 on the three splits of COCO. We do not observe a significant improvement in performance due to the augmentation.

3.3.6.2 Effect of backbone

We analyze the effect of backbone on our approach and show results for the normal convolution based (C4) and FPN based (FPN) backbones in Table 3.7. The first row corresponds to C4 while the second row corresponds to FPN. While the gap in performance is lower in most cases, we observe a significant improvement using the FPN on the low sparsity settings of all the split.

3.3.7 RPN Recall experiments for object discovery

PPM identifies foreground regions mistakenly assigned as background during training to avoid penalizing the network. To study the effect of PPM on novel [136] classes, we train a network using our approach on randomly chosen 6000 images of the COCO dataset, containing

Figure 3.5: Qualitative results showing the unlabeled regions identified by the PPM. The red boxes correspond to the available ground truth. A class agnostic NMS was performed on the regions and the result is shown in white.

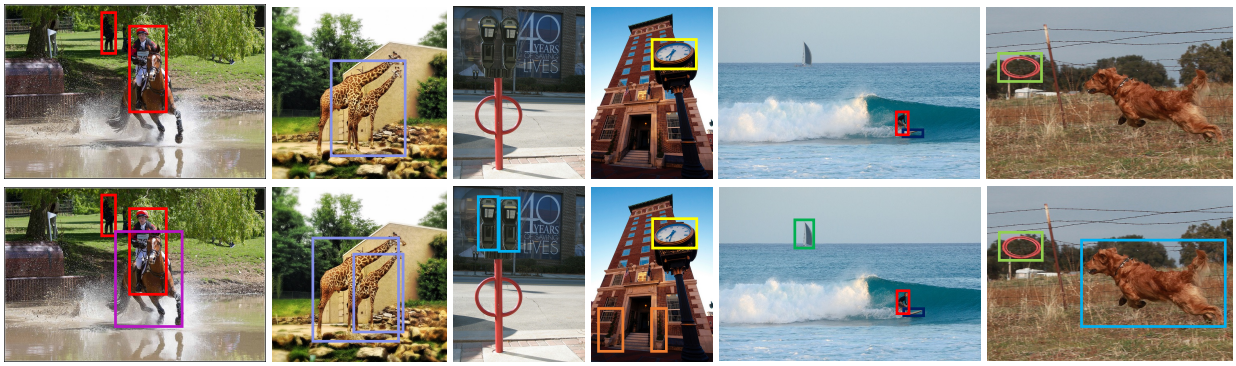


Figure 3.6: Qualitative results comparing the output of a model trained using available ground truths (top) to a model trained using our approach (bottom). Predictions with a class confidence score greater than 0.9 are shown. Red: Person, Cyan: Dog, Purple: Horse, Yellow: Clock, Green: Stop sign, Blue: Parking meter, Violet: Giraffe, Orange: Potted plant, Black: Surfboard, Dark green: Boat

annotations for only 20 classes of PASCAL-VOC. We evaluate the recall@0.5 of the RPN over the remaining 60 classes. A model trained using the standard technique on this dataset achieves a recall@0.5 of 77.46% and 29.06% on the known classes (20 categories) and unknown classes (60 categories) respectively. Our proposed approach, with PPM, achieves a recall@0.5 of 78.47% and 35.20% respectively. This ability to localize objects not seen during training can be beneficial for object discovery methods like [137, 138, 139, 140] which use RPN proposals to learn/discover new categories.

3.3.8 Qualitative Results

In Figure 3.5, we show the pseudo positives mined by PPM. In each figure, the red boxes correspond to the ground truth annotations and the white boxes correspond to the post NMS pseudo positive boxes mined by PPM. We observe that the PPM correctly mines proposals which correspond to missing object annotations. Without PPM, these regions will be used as negatives to the classifier resulting in a reduced class confidence score leading to a drop in mAP. We show the detection results of a model trained on the 50% Split-1 of our approach in Figure 3.6. The images on the top corresponds to the model trained using sparse annotations and the bottom image shows the output of our approach.

3.4 Discussion

We present SparseDet, a novel end-to-end system for Sparsely Annotated Object Detection (SAOD). We propose a simple yet effective technique for identifying the unlabeled regions using pseudo-positive mining and apply self-supervised loss on them. Through qualitative results we highlight the ability of PPM to mine pseudo-positives. We standardize the evaluation setup and show the effectiveness of our approach with an exhaustive set of experiments on multiple splits of SAOD. While we outperform existing state-of-the-art on all metrics and splits, we observe the gap in performance increases with sparsity demonstrating the short coming of methods that rely solely on pseudo-labeling. We propose a new benchmark, that evaluates the semi-supervised learning capabilities of SAOD approaches. We will release the data for the new benchmark along with all the SAOD splits and encourage researchers to evaluate future SAOD methods on these.

Table 3.8: Results on the splits released by authors of Co-mining. All methods use Res-50 FPN architecture.

Method	**Performance**		
	Easy	*Hard*	*Extreme*
Co-mining	35.40	31.80	23.00
Ours	**36.78**	**34.02**	**23.35**

Table 3.9: Comparison with on Split-2

Method	**Performance**		
	30%	*50%*	*70%*
Niitani *et al*.(Pseudo Label)	35.00	32.79	**29.03**
Wu *et al*.(Soft Sampling)	33.98	31.39	27.30
Ours	**35.94**	**33.13**	28.88

Table 3.10: Analysis of different augmentations used in this work.

Contrast	**Brightness**	**Saturation**	**Lighting**	**Bbox-Erase**	**Performance**
✗	✗	✗	✗	✗	42.76
✓	✗	✗	✗	✗	43.99
✗	✓	✗	✗	✗	43.81
✗	✗	✓	✗	✗	42.70
✗	✗	✗	✓	✗	42.95
✗	✗	✗	✗	✓	45.11
✓	✓	✗	✗	✗	44.03
✓	✓	✓	✗	✗	43.90
✓	✓	✓	✓	✗	43.91
✓	✓	✓	✓	✓	46.00

Table 3.11: Analysis on warm-up iteration for PPM.

Iteration	0	4000	9000	12000
AP	45.51	45.71	**46.00**	45.88

3.5 Appendix

3.5.1 Recall analysis

To study the effect of missing annotations on the RPN, we compute the recall of the RPN trained using a sparsely annotated dataset. For a model trained using all the annotations, the recall of the RPN is 0.83. As we drop annotations progressively from 30% to 70% the recall drops to 0.79. This shows that there is no significant drop in recall due to missing annotations.

3.5.2 Additional Experiments

3.5.2.1 Experiments on publicly released splits

Authors of Co-mining create a split similar to Split-4 but using COCO 2017 trainval set to perform ablation experiments. To the best of our knowledge, this is the only split released publicly for SAOD. We compare the performance of our approach with Co-Mining, which uses a

RetineNet architecture, on this split and report results in Table 3.8. We observe a similar trend as reported in the main paper and outperform Co-mining by 1.38, 2.22 and 0.35 percentage points on the *Easy*, *Hard* and *Extreme* subsets respectively.

3.5.2.2 Effect of τ_m

We ablate over different values of τ_m and report the results in Table 3.12 . With a low threshold, the ground truth will be contaminated with noisy predictions and a high threshold will leave out a few positive missing annotations. We observed that a value of 0.9 is a good tradeoff between quality and recall. We use a threshold of 0.9 for all the datasets unless otherwise stated.

3.5.2.3 Warm-up

As we rely on the RPN objectness score, we employ a warmup strategy which switches on the pseudo-positive mining (PPM) during training. In Table 3.11 we start the PPM at different iterations and report results. This experiment is performed on the COCO-mini ablation dataset. Our experiments show that for models which are trained for fewer iterations, starting at 9000 iterations worked the best. For longer experiments we allow a warmup of 30000 iterations.

Table 3.12: Ablation for τ_m

Threshold	0.8	0.9	0.95
AP	45.04	**46.00**	45.38

3.5.2.4 Additional ablations on C-RPN

We perform some additional analysis regarding C-RPN by using two sets of proposals, one from original and the other from augmented version of the image and combining them. This approach consists of $2\times$ more proposals than our approach and achieves an mAP of 45.25 (vs. **46.00** using ours). To make the comparison fairer, we generate half the number of proposals from each image and combine them (resulting in the same number of proposals as our approach) and this model achieves 45.46 (vs. **46.00** using ours) demonstrating the effectiveness of C-RPN.

3.5.2.5 Augmentation

The proposed approach process an input image and its augmented counterpart. For augmentation, a cascade of random contrast, brightness, saturation, lighting and bounding box erase are used. In Table 3.10, we analyze the effect of various augmentations on the performance of the model. This experiment was conducted on the same ablation set as the main paper. We observe that applying random saturation or random lighting alone are not as effective compared to other augmentations. Applying bounding box erase alone provides the most improvement. Finally, we achieve the best performance when we use all the augmentations. We randomly sample contrast, brightness, and saturation from $[0.5, 1.5]$. For lighting, a random scale of 1.2 was used. For bounding box erase, we randomly erase an area of of $[0.4, 0.7]$ at an aspect ratio of $[0.3, 3.3]$.

3.5.3 Additional details of splits and experiments

In Table 3.9, we compare our approach with Niitani *et al.*(Pseudo Label) and Wu *et al.*(Soft Sampling) using the same backbone for fair comparison. We outperform the best model (Pseudo

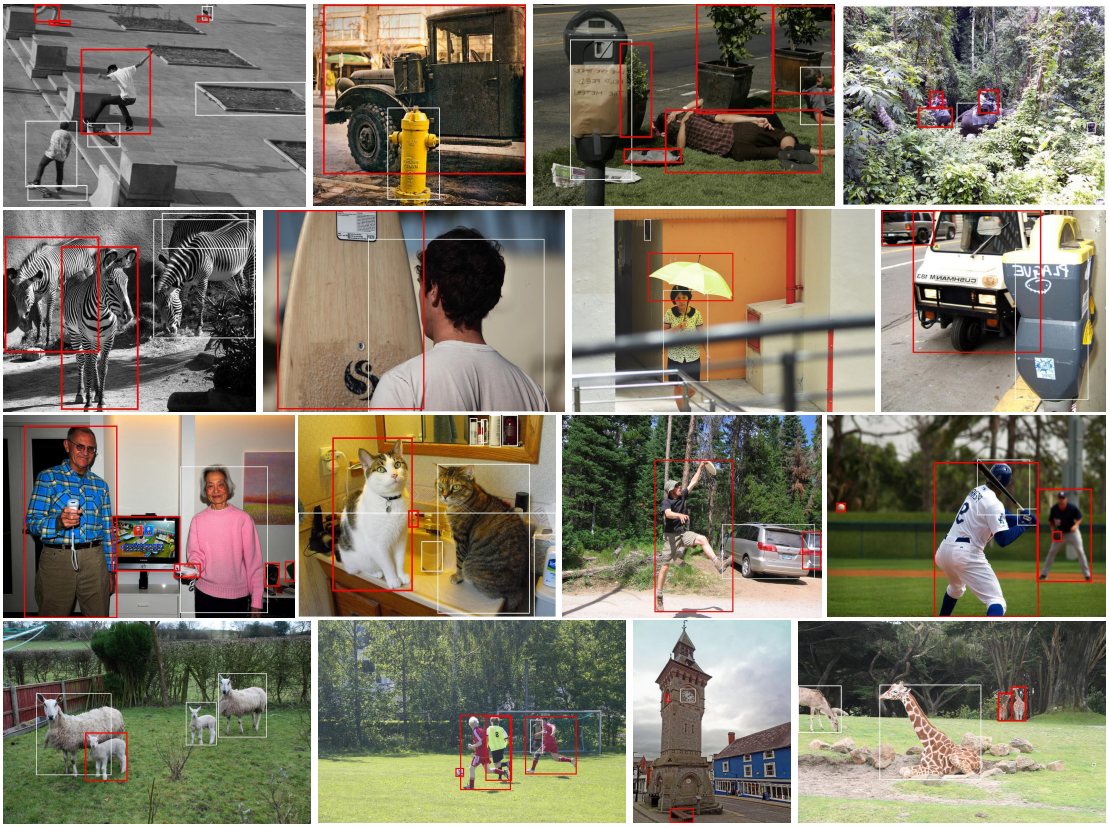


Figure 3.7: Figure showing regions (white) identified by the PPM step and available ground truth (red) during training.

Figure 3.8: Qualitative results comparing the output of a model trained using available ground truths (top) to a model trained using our approach (bottom). Predictions with a class confidence score greater than 0.9 are shown.

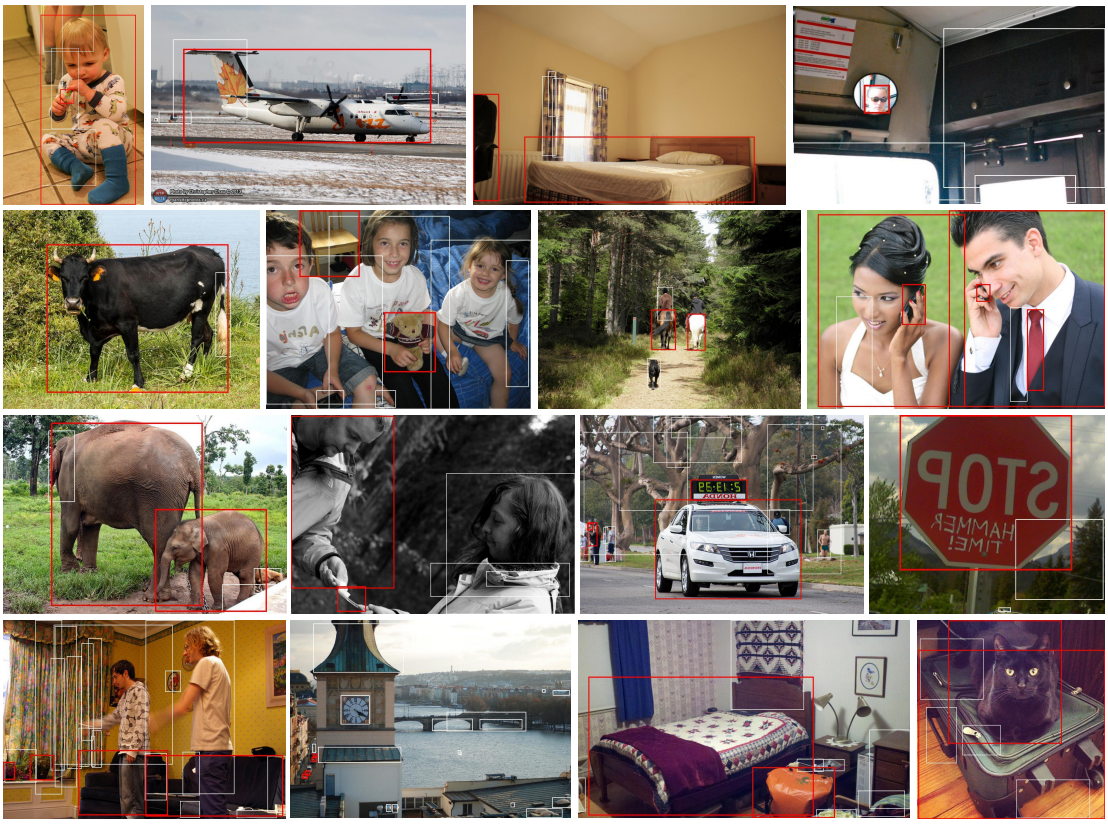


Figure 3.9: Failure cases of PPM.

Figure 3.10: Failure cases comparing the output of a model trained using available ground truths (top) to a model trained using our approach (bottom). Predictions with a class confidence score greater than 0.9 are shown.

Label) by 0.94, 0.34 points on the 30% and 50% splits respectively.

3.5.4 Results

In this section, we show some more results of the regions identified by the PPM in Figure 3.7 and a few failure cases in Figure 3.9. In Figures 3.7, 3.9 the red boxes correspond to the available annotations and the regions in white are the ones identified by PPM. Finally, we show detection results and failures of our approach in Figures 3.8 and 3.10 respectively.

Chapter 4: Towards Discovery and Attribution of Open-world GAN Generated Images

With an ever increasing number of GANs introduced each year, concerns about the malicious use of this technology, especially in the case of social media content [141, 142, 143] are increasing at an alarming rate which can have adverse impacts on public security and privacy. Therefore, a plethora of works have been proposed which focus on the real/fake detection task [3, 144, 145]. However, it is also important to focus on the problem of attribution, i.e. identifying the source of these images. Attributing images to their sources can potentially deter malicious organizations and hold them accountable by leading legal proceedings. Additionally, as GANs are becoming part of commercial services such as face animation applications, their popularity draws piracy and plagiarism [2] which is an attack on intellectual property. Therefore, it is pertinent to develop effective techniques to attribute images to specific sources.

To address this problem, [2, 146, 147] perform attribution for multiple GAN architectures and obtain high classification accuracies. However, they are limited to the closed-world setup as they attribute only to the GANs seen during training and are incapable of identifying unseen GANs. Such a setup is infeasible in practical scenarios where there are a large number of images belonging to sources not seen during training. This raises the question of whether we can discover these new sources and group together the set of images which are generated by them. We term this

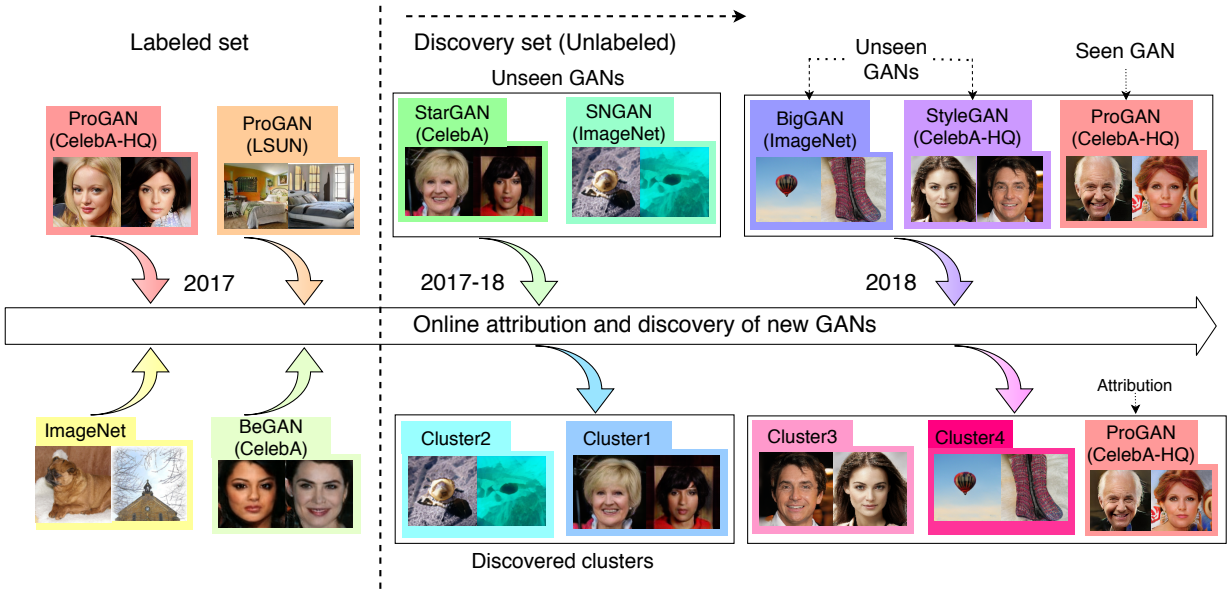


Figure 4.1: A plethora of GANs are released every year, and there could be a set of images that come from several unknown sources. Our approach is capable of discovering and attributing unknown GAN sources while requiring label supervision for only an initial small set of GANs. We attribute with high accuracies, seen GANs from a set of images as well as identify and cluster unknown GAN sources with high purities.

problem as "GAN discovery and attribution" as it involves attributing images to known sources as well as discovering unknown sources. This is a much more challenging and real world setup as the number of sources are unknown and keep increasing. Additionally, there can be a significant domain shift based on the dataset type of GAN generated images.

Many works such as [2, 3, 146] show that GANs leave unique artificial signatures in the images they generate. We exploit this information to implicitly identify signatures and cluster images belonging to unseen sources together while also attributing images to seen sources. We propose a novel iterative pipeline which utilizes a fixed set of images, labeled according to their corresponding sources, and perform GAN attribution and discovery on an unlabeled set of images. Our approach generalizes to an open-world setup where images in the unlabeled/discovery set are not restricted to be from the labeled class sources. Additionally, due to the iterative nature of our pipeline, we can continuously discover images from new GANs added to our discovery

set in an online manner. Our approach only requires labels for an initial set of images from real datasets and a few GANs trained on these real datasets with each real/GAN source representing a separate class. While we can discover unseen GANs trained on these real datasets, we additionally show through experiments that we can discover new real datasets and GANs trained on these new datasets as well without them being present in the initial labeled set.

Attribution and discovery in an open-world setup requires us to separate images belonging to seen sources during training from the unseen sources. We therefore introduce an explicit out-of-distribution (OOD) step using the deep network features to separate the images belonging to the two types of sources. We propose to incorporate the Winner Take All (WTA) hash [148] which, to the best of our knowledge, has previously never been used for OOD detection. Additionally, we obtain clusters for the OOD images and perform merge and refine steps to improve the grouping of the unknown GANs using 1-Nearest Neighbour (NN) graphs and kernel SVMs, respectively. We combine these components into a single unified pipeline which is executed iteratively for improving the features and clusters while attributing seen sources and discovering new GAN sources.

Through extensive experiments, we demonstrate the capability of our approach in an open-world setup. We show the efficacy of our approach to generalize to a wide range of dataset setups. We also analyze the importance of the various stages. Additionally, we provide an approach to apply our algorithm for the problem of real/fake image detection and show competitive results on a variety of dataset setups.

We summarize our contributions as follows: **1)** We introduce a new problem for discovering and attributing images from real and GAN sources in an open-world setup; **2)** We propose a novel iterative pipeline consisting of several components such as OOD detection, clustering and merge

and refine stages providing a strong benchmark for this task, and; **3)** We analyze the capability of our approach to discover GANs on a variety of dataset setups and also present several insights into the various stages of our pipeline.

4.1 Related Work

OOD Detection and Open Set Recognition: Several works [5, 149] have tackled OOD detection but require an OOD dataset for tuning hyperparameters, which is not possible as open-world knowledge is not known apriori. [150] removes this constraint but requires modification of the training setup to decompose confidence scores into two probabilities.

On similar lines is the task of open set recognition [151]. [152, 153, 154] use the Extreme Value Theory to discard unknown samples but require setting thresholds for reconstruction errors and/or probability values to detect OOD samples which requires careful tuning for each dataset. [155] provides a detailed survey of more works in this area.

Open World learning: While Open Set Recognition only rejects the unseen classes, Open World learning [156] also focuses on reasoning about the unseen classes. [157] tackles this problem using meta classifiers but are limited to the product classification problem. [4, 158, 159, 160] also focus on a similar problem but require the unlabeled set to only contain unseen classes and knowledge about number of unseen classes in some cases.

Rank correlation: [148] compute the WTA hash which are ordinal embeddings providing a highly non-linear sparse transformation of the feature vector. [161] use this hashing algorithm for performing fast large scale object detection. To the best of our knowledge, no work utilizes ranking based measures for OOD detection.

Clustering: Clustering is a highly explored field yet there is no one-size fits all solution. [162] use a first Nearest Neighbours (1-NN) graph to perform parameter free clustering. Inspired from their work, we use a similar 1-NN graph for our merge step. [163] perform K-means clustering on a network's features and retrain the network using the cluster labels as pseudo-labels. Our approach partly involves this setup but contains several other components such as OOD detection and merge and refine steps. Spectral clustering [164, 165, 166] is another common approach but requires eigenvalues for a large Laplacian which is not tractable for large datasets, as is our case. Another common direction is training a deep network [167, 168, 169, 170, 171] which learns embeddings/clusters based on minimizing an objective function. However, these require careful training so as to not diverge while learning the features in an unsupervised manner.

Real/fake detection and GAN attribution: A plethora of works [3, 142, 143, 144] exist for the problem of real/fake detection but are only limited to this binary classification problem and are not directly applicable to GAN attribution and discovery. [2, 3, 146] tackle this problem but are, however, limited to the GANs that they train on and fail to generalize in an open world setup. [172] propose a more dynamic approach to incrementally include GANs for attribution but require clean datasets with images coming from only a single GAN source which does not hold in practice, as images could be generated from multiple sources. To the best of our knowledge, there exists no work dealing with open-world GAN discovery and attribution which is a much harder task than just real/fake detection or closed set GAN attribution.

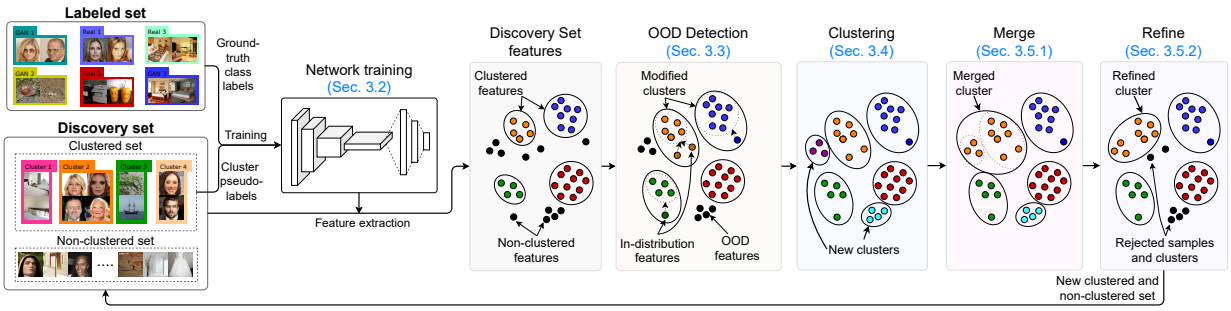


Figure 4.2: Illustration of our algorithm, where we iteratively discover new classes and retrain our network using them as pseudo-labels.

4.2 Proposed Approach

4.2.1 Overview

In this section, we briefly describe our approach as shown in Fig. 4.2. Our initial labeled set consists of n_s images corresponding to the seen classes and is denoted by $\mathcal{I}_s = \{I_{s_1}, I_{s_2}, ..., I_{s_{n_s}}\}$, and their ground truth class labels, denoted by $\mathcal{Y}_s = \{y_{s_1}, y_{s_2}, ..., y_{s_{n_s}}\}$. The discovery set consists of n_t unlabeled images, from both seen and unseen classes, and is denoted by $\mathcal{I}_t = \{I_{t_1}, I_{t_2}, ..., I_{t_{n_t}}\}$. Our pipeline proceeds iteratively, and at any point in the pipeline, our discovery set is partitioned into $\mathcal{I}_c$ and $\mathcal{I}_n$. $\mathcal{I}_c$ is a set of n_c clustered images with predicted labels $\hat{\mathcal{Y}}_c$ while $\mathcal{I}_n$ is a set of images which could be potentially clustered in future iterations.

In each iteration, we improve the predicted labels in the clustered set ($\mathcal{I}_c$) and add new samples from the non-clustered set ($\mathcal{I}_n$) into the clustered set. We do this via several stages using algorithms or tools which have previously not been applied for the specific tasks. We also combine the various stages in a unified manner for iteratively improving the features and clusters.

Network training: Our network consists of a feature extractor $f(\cdot)$ and classifier $g(\cdot)$. We train the network in a supervised manner using the two sets of images and labels, labeled set $(\mathcal{I}_s, \mathcal{Y}_s)$

and clustered set $\left(\mathcal{I}_c, \hat{\mathcal{Y}}_c\right)$.

Out-of-distribution detection: We use $f(\cdot)$ to extract features for $\mathcal{I}_c$ and $\mathcal{I}_n$ and perform OOD detection. This stage predicts samples from $\mathcal{I}_n$ to be in-distribution or OOD with respect to the clusters in $\mathcal{I}_c$. The in-distribution samples are classified using the classifier and attributed to $\mathcal{I}_c$ with the corresponding predicted labels.

Clustering: We use the K-means algorithm to overcluster the remaining samples in $\mathcal{I}_n$. These clusters are then added to the clustered set $\mathcal{I}_c$ with a new set of labels based on the cluster labels. At the end of this stage all samples have a predicted label and the non-clustered set, $\mathcal{I}_n$, is empty.

Merge and refine: To deal with overclustering we perform merge and refine operations. Specifically, coherent clusters are merged to reduce the number of clusters. This reduces the purity of the clusters and hence a refine operation is performed which throws away impure clusters, or samples likely to not have belonged to their existing clusters. The rejected samples are added to the non-clustered set $\mathcal{I}_n$. At the end of this stage, we have a new clustered set, $\mathcal{I}_c$ along with its predicted labels $\hat{\mathcal{Y}}_c$, and non-clustered set, $\mathcal{I}_n$. The four steps described above are then repeated.

We now describe each of the steps enumerated above in detail.

4.2.2 Network Training

This stage involves training the network using the cluster labels $\hat{\mathcal{Y}}_c$ corresponding to $\mathcal{I}_c$ and $\mathcal{Y}_s$ corresponding to $\mathcal{I}_s$ in a supervised manner. The network consists of a feature generation network $f(\cdot)$ parameterized by θ_f, constructed using an off-the-shelf CNN followed by a few fully connected layers to reduce the dimensionality. The classification part of the network $g(\cdot)$ parameterized by θ_g involves a fully connected layer followed by the softmax function.

The parameters of the network θ_f, θ_g are optimized as per the following expression:

$$\min_{\theta_f,\theta_g} \left[\frac{1}{n_c} \sum_{i=1}^{n_c} \mathcal{L}\left(g_{\theta_g}(f_{\theta_f}(I_{c_i})), \hat{y}_{c_i}\right) + \frac{1}{n_s} \sum_{j=1}^{n_s} \mathcal{L}\left(g_{\theta_g}(f_{\theta_f}(I_{s_j})), y_{s_j}\right) \right], \tag{4.1}$$

where $\mathcal{L}$ is the cross-entropy loss, I_{c_i} and $\hat{y}_{c_i}$ are the i^{th} images and labels from $\mathcal{I}_c$ and $\hat{\mathcal{Y}}_c$ respectively while I_{s_j} and y_{s_j} are the j^{th} images and labels from $\mathcal{I}_s$ and $\mathcal{Y}_s$ respectively. Subsequent to network training, we use the feature generation network to extract the features $\mathcal{X}_c$ and $\mathcal{X}_n$ corresponding to the clustered set of images $\mathcal{I}_c$, and non-clustered set of images $\mathcal{I}_n$, respectively.

4.2.3 Out-of-distribution detection

We utilize the WTA hashing algorithm proposed by Yagnik *et al.*[148] who show that ordinal representations of feature vectors provide strong nonlinear transformations and demonstrate their algorithm's capability on downstream tasks, such as similarity search and classification. They show that such rank correlation measures are robust to noise unlike cosine or Euclidean based distances. Additionally, Euclidean/cosine based distances are highly sensitive to thresholds used for OOD detection which would require careful hyperparameter tuning for different dataset setups.

The WTA hash maps a d dimensional feature vector $\boldsymbol{x}$ to a H dimensional vector $\boldsymbol{x}_H$ with elements lying in $[K]$.

Using this hash for each feature vector, we then represent the distance between any two feature vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, as $d(\boldsymbol{x}, \boldsymbol{y})$, which is the Hamming distance between their corresponding hashes. For each class in set $\hat{\mathcal{Y}}_c = \{\hat{y}_i \in [N],\ i \in [n_c]\}$ (n_c is the number of samples in the clustered set, N is number of clusters), we obtain OOD detectors in the following manner: For

a cluster with cluster label $j \in [N]$ and for a feature sample $i \in [n_c]$ in the non-clustered set represented by $\boldsymbol{x}_{n_i} \in \mathcal{X}_n$, we compute the distance of $\boldsymbol{x}_{n_i}$ from each sample in the cluster j. We then average these sample distances to get the distance of sample $\boldsymbol{x}_{n_i}$ from cluster j, *i.e.*,

$$d_j(x_{n_i}) = \frac{1}{N_j} \sum_{k=1, y_{c_k}=j}^{n_c} d(\boldsymbol{x}_{n_i}, \boldsymbol{x}_{c_k}), \tag{4.2}$$

where N_j represents the number of samples in cluster j. The detector then classifies $\boldsymbol{x}_{n_i}$ as an in-distribution sample of class j if $d_j(x_{n_i}) < t_j$ for a threshold t_j for class j. The threshold t_j is computed using the intra cluster distances for each cluster j and setting a high percentile of these distances as the threshold. By doing so, the algorithm learns different thresholds for different clusters and is controlled only by a single percentile scalar which generalizes across different dataset setups.

A test sample, $\boldsymbol{x}_{n_i}$ is classified as an OOD sample to $\mathcal{X}_c$, if all of the detectors for the clusters classify it as OOD. All in-distribution samples are classified using our classifier and their corresponding labels lie in $\hat{\mathcal{Y}}_c$. The samples are subsequently added to $\mathcal{I}_c$.

4.2.4 Clustering

We now overcluster samples remaining in $\mathcal{I}_n$ by running K-Means on the feature set $\mathcal{X}_n$. We form a high number of clusters in order to get clusters with high purity. Once the clusters are obtained, they are added to the clustered set $\mathcal{I}_c$. Their new labels, corresponding to the cluster labels, are added to $\hat{\mathcal{Y}}_c$.

At the end of this stage, no samples remain in the non-clustered set. More importantly, as we generate a large number of clusters, it makes the clustered set highly fragmented. In order

to reduce the number of clusters and improve the purity of the clusters we perform a merge and refine step as explained in the following section.

4.2.5 Merge and refine

Overclustering results in a highly fragmented cluster set which could belong to the same class. To deal with this, a merge step is performed. Anything less than an ideal merge step results in impure clusters. To improve the purity a refine step is also performed. We discuss these in detail below.

4.2.5.1 Merge

We merge clusters in $\mathcal{I}_c$ using a 1-Nearest Neighbour graph. We obtain centroids, $\boldsymbol{u}_j$, for each cluster $j \in [N]$ (N is the number of clusters) by averaging the features of all samples in the cluster. Using the hashing described in Section 4.2.3 for each centroid, we define the distance between two centroid feature vectors $\boldsymbol{u}_i$ and $\boldsymbol{u}_j$, $d(\boldsymbol{u}_i, \boldsymbol{u}_j)$, as the Hamming distance between their corresponding hashes $\boldsymbol{u}_{iH}$ and $\boldsymbol{u}_{jH}$. We use the centroid distances between every pair of clusters to create a directed 1-Nearest Neighbour graph with each node representing a cluster centroid. A directed edge is present from one node to another if the latter node is the nearest neighbour centroid of the former node. Strongly connected components are computed for this graph and each connected component in the graph is considered to be a merged cluster. This stage generates a new set of labels, $\hat{\mathcal{Y}}_c$, for the clustered set $\mathcal{I}_c$.

Table 4.1: List of GANs trained on the corresponding 4 real datasets used in our labeled and discovery set. Note that the same GAN can be trained on multiple datasets.

Dataset	**Labeled GANs**	**Discovery GANs**
CelebA[173]	StarGAN[174], AttGAN[175]	StarGAN, BEGAN[176], ProGAN[177], SNGAN [178], AttGAN, MMDGAN[179], CramerGAN[180]
CelebA-HQ [177]	ProGAN, StyleGAN[181]	ProGAN, StyleGAN, ResNet19[182]
ImageNet [66]	BigGAN[183], S3GAN[184]	BigGAN, S3GAN, SNGAN
LSUN Bedroom [185]	ProGAN, MMDGAN	ProGAN, MMDGAN, SNGAN

4.2.5.2 Refine

As the merge step is not ideal, it reduces the average purity of the clusters. In order to increase it, a refine step is performed to remove impure samples from each cluster. As the ground truth labels are unknown, SVM classifiers are leveraged to obtain a proxy measure for purity.

[186, 187] show that weak SVM classifiers can be fit to a single positive instance with the remaining samples as negatives. Therefore, we use this formulation of weak classifiers that can fit to the majority class distribution of a cluster and mark the samples which do not belong to the majority class as negatives.

For each cluster $j \in [N]$, an SVM classifier, Q_j, is trained in a one-vs-all manner, where the positive samples belong to cluster j while the rest of the samples in the clustered set are negative samples. After training Q_j, we use the SVM to predict the labels for samples in cluster j as positive and negative. The samples which are predicted negative are then rejected and added back into the non-clustered set $\mathcal{I}_n$. If the percentage of predicted positive samples by Q_j in cluster j is below a threshold ϵ, the entire cluster is discarded and all the samples are added to $\mathcal{I}_n$.

Additionally, some refined clusters might have very few samples and the class distribution

for training the network in the next iteration could become long tailed. In order to avoid this issue, we threshold clusters based on their sizes and discard those below a size threshold τ into $\mathcal{I}_n$.

After the refine step we have a new set of clustered images with their corresponding pseudolabels. These are used along with the seen class train data $\mathcal{I}_s$ in order to train the network for the next iteration.

4.2.6 Cluster set initialization

The start of every iteration of our pipeline requires a clustered set $\mathcal{I}_c$ along with the seen labeled set $\mathcal{I}_s$. For the first iteration, as we do not have any pseudolabels for the discovery set $\mathcal{I}_t$, we train our network using only the set $\mathcal{I}_s$ and their corresponding ground truth labels $\mathcal{Y}_s$. Our OOD detection step then determines whether images in $\mathcal{I}_t$ belong to the seen classes $\mathcal{Y}_s$ or not. In-distribution samples are classified and are added to the clustered set $\mathcal{I}_c$ while OOD samples are added to $\mathcal{I}_n$. At the end of this stage, we now have a clustered and non-clustered set for the discovery set images. The rest of the stages of our pipeline, *i.e.*, K-Means Clustering, Merging and Refinement proceed as explained in the previous sections using the initialized $\mathcal{I}_c$ and $\mathcal{I}_n$. The refine step then produces a set of images in the clustered set with their corresponding cluster labels as pseudo-labels which are used to train the network for the next iteration. Additionally, at every iteration t, the feature extractor is initialized with the weights of the previous iteration $t-1$. The classifier is replaced with a new linear layer with weights randomly initialized as number of classes, which is dependent on number of clusters N, change across iterations. The algorithm then proceeds for a few iterations until fraction of undiscovered samples fall below a

small threshold.

4.3 Experiments

We now evaluate our approach on real world dataset setups while providing detailed analysis of the several components of our pipeline. In Section 4.3.1, we describe the implementation details. Our labeled dataset consists of images from 4 real datasets as well as from certain GANs trained on these real datasets as shown in Table 4.1. Together, they make up 12 classes in the labeled set. Our discovery set consists of additional images from these 12 classes as well as from 8 unseen GANs as shown in Table 4.1 making up a total of 20 classes. We use, by default, this dataset for all our experiments unless mentioned otherwise. Note that the same GAN trained on different datasets corresponds to different classes. Section 4.3.2 shows extensive comparisons with other related works on GAN attribution and real/fake image detection. Section 4.3.3 provides several insights into our algorithm and also analyzes several components of our pipeline. Subsequently, we examine the results of our pipeline on varying dataset setups. Section 4.3.4.1 shows an analysis of number of GANs needed in our labeled set to reliably discover new GANs in the discovery set. Section 4.3.4.2 changes number of unseen real datasets as well as corresponding GANs in the discovery set and shows the effectiveness of our approach to discover these new classes.

4.3.1 Experimental details

For our feature extractor, we use the standard ResNet-50 [188] backbone. We add 3 fully-connected layers to reduce the dimensionality of the feature vector to 128. Another fully con-

nected layer is used as the classification head on top of the feature extractor. The full network is trained in a supervised manner and using cross entropy loss. Every image is resized and center cropped to 256×256 except when specified otherwise. We use a batch size of 256 for our training for each iteration of the pipeline. The weights are optimized using the Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and a fixed learning rate of 0.0001 throughout our training. For the first iteration, we train our network for 50 epochs, while for subsequent iterations we train for 100 epochs, as the network takes longer to converge with additionally discovered samples with noisy pseudo labels. For our OOD detection step using WTA hash described in Section 4.2.3, we use $H = 2048$ hashes and a window size of $K = 2$. Our clustering stage uses the K-Means algorithm for 500 clusters initialized using K-Means++ [189]. For the refine stage, we train SVMs with the RBF-kernel. We set the threshold, $\epsilon = 0.5$, for dropping a cluster, as described in Section 4.2.5.2. To avoid training on clusters with very few samples, we discard clusters with less than 100 members.

Metrics and analysis: We evaluate our pipeline on 2 clustering metrics. We use Average Purity as a metric for evaluating the overall purity of our clusters with respect to the true labels of the discovery set. We also use Normalized Mutual Information (NMI), which is another commonly used clustering metric. At various stages or iterations of our pipeline, a small fraction of the discovery set samples remain non-clustered and in order to provide a fair evaluation across different experiments/baselines we attribute all the non-clustered samples to their nearest clusters and evaluate on the full discovery set, unless mentioned otherwise.

Table 4.2: Comparison of our method with baselines derived from [2, 3, 4]. We try two fixed setups for number of clusters $k = 20, 500$ and finally let our approach discover the suitable number of clusters $k = 209$. Compared to the 2 baselines, we obtain the highest Average Purity and NMI when number of clusters $k = 209$. Ours [only §3.2] corresponds to a single iteration of network training and clustering. The fully supervised setup is the upper bound when all classes are seen.

Method	$k = 20$		$k = 500$		$k = 209$	
	Avg. Purity	NMI	Avg. Purity	NMI	Avg. Purity	NMI
Yu *et al*. [2]	0.656	0.706	0.759	0.518	0.734	0.554
Han *et al*. [4]	0.680	0.709	-	-	-	-
Wang *et al*. [3]	0.710	0.759	0.857	0.575	0.840	0.624
Ours [only §3.2]	0.661	0.743	0.814	0.561	0.795	0.609
Ours	-	-	-	-	**0.861**	**0.724**
Fully supervised	0.928	0.929	0.996	0.658	0.997	0.728

4.3.2 Benchmark Evaluation

As there exists no prior work dealing with open-world GAN discovery, we provide baselines by modifying recent works involving GAN attribution [2] and real/fake image detection [3]. We additionally include the recent approach of [4] which deals with novel category discovery.

Yu *et al*. [2] deals with GAN attribution in a closed-world setup and hence cannot be directly incorporated to our problem setup. Therefore, we train their network on our labeled set and obtain features for our discovery set. We cluster the features using K-Means for 3 different values of k. $k = 20$ corresponds to the true number of classes in our test set while $k = 500$ corresponds to an overclustered regime. $k = 209$ represents the number of clusters our algorithm returns at the end of 4 iterations. We compare across multiple values of k as Average Purity and NMI are known to be sensitive to number of clusters.

Wang *et al*. [3] tackles real/fake detection and again cannot be directly used in our problem setup. Therefore, we modify their classification head to be multiclass and train their network

on our labeled set using their training and preprocessing strategies and extract the features for our discovery set. We provide three similar baselines by performing clustering similar to the baselines generated from [2].

Han *et al*. [4] discover novel visual categories but require the discovery set to only contain unseen classes. We therefore use our anomaly detection approach on their features to separate out the seen and unseen classes whose cluster assignments are then predicted separately using their approach. As they require knowledge of number of unseen classes for their predictions, we compare with the $k = 20$ setup which corresponds to the true number of classes.

Finally, we provide a baseline for our approach by performing network training and clustering the feature space into $k = 20, 500, 209$ clusters. We also provide an upper-bound for our approach using a fully supervised case where the labeled set consists of images from all classes in the discovery set and perform clustering on the generated features.

The results for these comparisons are provided in Table 4.2. Our algorithm achieves the highest Average Purity and NMI compared to all other baselines for the case of $k = 209$. For $k = 20, 500$, [3] outperforms a single iteration of network training and clustering because it does not involve OOD detection, merge or refine for this comparison. However, at the end of 4 iterations, for the case of $k = 209$, we significantly outperform all baselines in terms of both Average Purity and NMI. The fully supervised approach provides an upper bound for all 3 cases. Note that we do not compare across number of clusters as Average Purity increases in general with more clusters while NMI decreases.

Table 4.3: We analyze the effect of the various stages of our pipeline. The number of clusters in the merge step decreases with negligible drop in Avg. Purity and increased NMI. The Refine step further increases the NMI and Avg. Purity by a big margin for the discovered samples. Note that the numbers corresponding to all samples in the refine step are included for the sake of fair comparison but are not actually computed by our approach.

Stage	No. of clusters	Avg. Purity	NMI
Clustering	512	0.793	0.682
Merge	391	0.792	0.689
Refine (Discovered)	111	**0.849**	**0.838**
Refine (All)	111	0.772	0.720

Table 4.4: We evaluate our algorithm over multiple iterations. Avg. Purity, NMI and % of discovered samples progressively increases.

Iteration	Avg. Purity	NMI	% Samples Clustered	Sources Discovered
1	0.772	0.720	72.5	16/20
2	0.853	0.724	88.8	20/20
3	0.861	0.724	92.3	20/20
4	**0.861**	**0.724**	**93.7**	**20/20**

4.3.3 Ablation Study

We evaluate the effect of Clustering, Merge, and Refine stages in the first iteration of our pipeline. The results are summarized in Table 4.3. Note that the Average Purity drops only slightly in the merge step while the number of clusters drop significantly demonstrating the effectiveness of the 1-NN merge step explained in Section 4.2.5. From the merge to the refine step, Average Purity drops for the full discovery set as many samples remain undiscovered and we evaluate the metric over the full discovery set by naïvely attributing them to the nearest cluster. However, the metrics evaluated on only the discovered samples increase significantly which shows that SVMs can identify the pure clusters and samples while rejecting the impure ones.

Next, we evaluate our pipeline over multiple iterations. We show the results in Table 4.4.

The pipeline discovers only a small fraction of images and GANs in the first iteration while in subsequent iterations, more samples are added to the clustered set and more GANs are discovered. Average Purity and NMI both increase or remain constant over the four iterations which shows the effectiveness of our approach to discover as well as improve clusters. Our OOD stage obtains an accuracy of 86.97% for seen classes, 99.87% for unseen classes and 92.87% overall. The high unseen class accuracy is because of setting a lower threshold to reduce false negative errors which do not get corrected in subsequent stages.

4.3.4 Varying dataset setups

In this section, we provide an analysis by varying the dataset setups based on number of GANs per real dataset in the labeled set or on adding new real datasets and GANs trained on them in the unlabeled set.

4.3.4.1 Effect of number of GANs per dataset

We answer the question of how many GANs per dataset are needed in our labeled set to reliably discover new ones in our discovery set. We have 3 labeled dataset setups: **1)** Our first setup consists of 4 real datasets: CelebA, CelebA-HQ, ImageNet and LSUN-Bedroom with no GANs; **2)** In addition to the 4 datasets in the first setup, our second setup has 4 GANs: StarGAN, ProGAN, BigGAN and MMDGAN trained on the respective datasets; **3)** In addition to the previous setup, we have 4 more GANs per dataset: AttGAN, StyleGAN, S3GAN and ProGAN.

In order to fairly evaluate the 3 setups, we use a common discovery set consisting of all the

Table 4.5: Varying number of GANs per dataset. We obtain the best metrics with the maximum number of GANs per dataset although discovering fewer samples compared to the first setup.

# of GANs	Avg. Purity	NMI	% Samples Clustered	Sources Discovered
0	0.497	0.559	**99.78**	8/12
1	0.897	0.772	94.48	11/12
2	**0.954**	**0.789**	95.98	**11/12**

Table 4.6: Effect of adding new datasets and GANs trained on new datasets at test time. (*) provides the corresponding comparison when the real datasets are present in the labeled set (Sec. 4.3.4.2)

Test Set	Purity	NMI	Sources Discovered	# of Clusters
New Real	0.942	0.813	14/16	103
New Real*	**0.989**	**0.989**	**15/16**	56
New GANs	**0.976**	0.828	**16/16**	105
New GANs*	0.95	**0.835**	15/16	87
New Real + New GANs	0.850	0.730	**20/20**	141
New Real + New GANs*	**0.977**	**0.856**	19/20	128

classes in the second setup. Additionally, we have a set of GANs not present in all 3 labeled sets, namely, BEGAN, ResNet19 (from CompareGAN [182]), SNGAN and CramerGAN corresponding to the 4 real datasets. The results are summarized in Table 4.5. Due to most information being present in the labeled set, the third setup performs best on both Average Purity and NMI. Despite the second setup having only a single GAN per dataset, it performs fairly well on the two metrics. On the other hand, the first setup, which does not have any GANs in the labeled set, fails to discover new ones as it cannot see any GAN-related artifacts in the labeled set and thus fails to discriminate based on this during discovery.

4.3.4.2 Discovering new dataset images

In an open-world setting, the discovery set may contain images from new real datasets not seen in the labeled set along with GAN generated images corresponding to these datasets. To see whether the proposed approach can handle these situations we perform experiments covering 3 setups. Each setup uses the default labeled set in Table 4.1 but additional classes in the discovery set as follows: **1)** New real datasets: New real datasets namely DTD [190], FashionGen [191], and Night and Shoes datasets (from Pix2Pix [192]); **2)** GANs on new real classes: New GANs trained on the four new real world datasets, namely, ProGAN on DTD, DCGAN [193] on FashionGen, and a separate Pix2Pix on Night and Shoes datasets; **3)** New Real + New GANs: A combination of GANs and real datasets from the previous two setups.

In order to provide a benchmark for comparison, we show the performance when the four real datasets are in the labeled set (marked with a *). The results are shown in Table 4.6. In the first setup the goal is to discover new dataset sources. Our approach discovers most of the sources with high Purity and NMI, although it's performance is lower than the benchmark as expected because the labeled set for the benchmark contains all the classes present in the discovery set. In the second setup, our method discovers all unseen GANs even though they are trained on unseen datasets unlike the benchmark which does slightly worse in terms of Avg. Purity and number of GANs discovered likely because of the reduced number of final clusters. The third setup is more challenging due to the addition of both unseen datasets and GANs trained on them to the discovery set. However our approach discovers all unseen sources with reliable Average Purity and NMI while its corresponding benchmark does not discover all sources possibly because it restricts itself to lesser but purer clusters with higher NMI.

Table 4.7: Evaluation of our algorithm in an online setup. We have 17 sources in our initial discovery set and add 3 sources each at iteration 3 and 5 causing an initial drop in results. The pipeline eventually performs better after training on the new samples.

Iteration	Avg. Purity	NMI	% Samples Clustered	Sources Discovered
1	0.846	0.826	89.39	15/17
2	0.916	0.798	92.42	16/17
3	0.805	0.771	95.74	18/20
4	0.805	0.744	96.87	19/20
5	0.731	0.716	95.68	22/23
6	0.802	0.705	95.36	22/23

4.3.4.3 Online discovery

Here we extend our approach to an online setup where new GANs are added to the discovery set in an online fashion based on the chronological order they were published. Our setup consists of 9 GANs from 4 real sources in our labeled set and 4 new GANs in the discovery set. We additionally introduce 2 sets of 3 GANs each in an online fashion. We show our results in Table 4.7. We train our setup for 2 iterations with the initial discovery set of 17 sources. It can be seen that Average Purity increases in the second step and it also discovers an additional GAN source. When new GANs are introduced in iterations 3 and 5, the performance drops as the network is not trained on the new classes. However, after a single iteration the Average Purity increases significantly and NMI drops only slightly even though number of clusters increase. At the end of 6 iterations, we discover all the GAN sources added on the fly, except one. This shows that our approach works in an online setting, continuously discovering new GANs iteratively.

Figure 4.3: Samples from clusters discovered by our approach for unseen GANs with the majority class in parenthesis. It can be noticed that they are not just focusing on the object structure and semantics rather the underlying source.

4.3.5 Real/Fake detection

We now apply our method to the common problem of real/fake detection. We use the binary classification model from [3], but trained on our labeled set and use majority voting to mark a cluster and all its constituent images as real or fake. We compare this with using the model directly on all samples and compare the performance in Table 4.8 for our original setup and for the three setups defined in Sec. 4.3.4.2. We observe that in most settings, we outperform the standard predictions which are evaluated sample-wise. We attribute it to the fact that the clustering is able to correct model's mistakes as it groups samples according to the source. As cluster assignments are less accurate due to increased difficulty of the final setup, our performance is lower but nevertheless, competitive with [3].

4.3.6 Qualitative analysis of clusters

We visually inspect a few clusters generated by our method to see whether they focus on the semantic information or the GAN source. To this end we visualize random images from some of the highly pure clusters corresponding to unseen GANs trained on ImageNet, LSUN-Bedroom, CelebA and CelebA-HQ. As evident from Fig. 4.3 the cluster in the case of SNGAN-ImageNet does not seem to be object-specific, while the cluster for SNGAN-LSUN does not

Table 4.8: We evaluate the real/fake detection accuracy (%) using the clustering obtained from our network.

Approach	Original	New Real	New GANs	New Real + New GANs
Wang *et al.* [3]	92.56%	87.35%	98.42%	**89.09%**
Ours	**98.62%**	**89.84%**	**99.10%**	83.33%

focus on specific room decor, lighting conditions, layout etc. Similarly, clusters corresponding to the face datasets seem to be focusing on the GAN source rather than specific facial attributes like expression, orientation, age etc.

4.4 Discussion

We proposed a new problem of open-world GAN discovery and attribution. We presented an iterative approach to discover and attribute images from multiple GAN sources in a discovery set. Our framework discovers and groups GANs not seen during training by implicitly focusing on GAN-based fingerprints. We show ablation studies for the different components of our pipeline. We also show the generalization of our approach to various dataset setups and its extension to an online setting. As there have been no works addressing this problem, we compare with several baselines based on state-of-the-art related works and provide a strong benchmark for this task. Even though our approach works in an online setup, network training is an expensive step for each iteration. One potential direction for future work is to utilize approaches from continual learning literature [194] for faster training, to learn in a never-ending setup discovering new GANs on-the-fly. We hope, given the general formulation of the stages, our framework is utilized for other similar tasks as well. To facilitate such exploration of different scenarios we plan to release the toolset we have developed for our work to bolster future research in this area.

4.5 Appendix

4.5.1 Additional experimental details

For our feature extractor, we use 3 fully-connected layers. Each fully connected layer is followed by the ReLU activation unit and Dropout with a drop probability of 0.5 during train phase for regularization. The first layer maps the input 2048 dimensional vector to 512 dimension. The second layer maintains the number of activation units at 512 while the third one downsamples it to 128 which is the final dimension of the feature vector we use for subsequent stages. For all our experiments in the supplementary, we train on 128×128 sized images. We set a percentile threshold of 0.9 for our out-of-distribution detection stage. Our cluster merging algorithm using the 1-Nearest Neighbour Graph is a 2 staged setup which initially merges the newly obtained clusters from K-Means in the previous stage and then merges the entire clustered set. We adopt this 2 staged setup as K-Means overclusters the discovery set and requires merging before merging with the clusters in the clustered set. For training SVMs, we use the GPU-accelerated library of ThunderSVM [195].

4.5.2 Additional dataset details

Table 4.9 summarizes the class-wise train and test splits used across our experiments. We use a variety of images for multiple image sources to more closely simulate a real world setup. Note that some train images are not used depending on the dataset setup where the image sources could only belong in the discovery set.

Our online dataset setup defined in Section. 4.4.3 of the paper consists of GANs in the chrono-

logical order they were published or introduced. We have an initial labeled and discovery set as defined in Table 4.10. After running our pipeline for 2 iterations on this set, we add 3 more GANs to the discovery set: BigGAN and SSGAN [196], both trained on ImageNet, StyleGAN trained on CelebA-HQ. We run our pipeline for 2 more iterations on the new discovery set, and add 3 more GANs: ResNet19 and StarGAN-v2 [197], both trained on CelebA-HQ, S3GAN trained on ImageNet. This is followed by 2 more iterations of network training resulting in a total of 6 iterations for the full online setup. The numbers are as reported in Section 4.4.3 of the main paper.

4.5.3 Additional baseline comparisons

Section 4.2 of the paper provides comparisons with baselines derived from the works of [2] and [3] by training their methods on our dataset in a multiclass manner. We additionally provide baselines by using features from the pretrained models provided by them which were trained on their datasets. We provide results by performing K-means clustering on the features for $k = 20$ and $k = 500$ similar to the baselines derived in Section 4.2. The results are shown in Table 4.11 (denoted by *). It can be seen that the features don't generalize across datasets and does worse than the baselines reported in the paper on both the metrics of Average Purity and NMI. Also, as [3] primarily deals with only real-fake classification, we train their method on our dataset but only on the binary real-fake classification task and extract their features. We show the results based on clustering the features for $k = 20$ and $k = 500$ and reporting results in Table 4.11 (denoted by †). We see that features generated from the binary classification problem do worse than the multiclass case. This is because the binary classification problem only discriminates between

Table 4.9: List of GANs trained on the corresponding 4 real datasets used in our labeled and discovery set. Note that the same GAN can be trained on multiple datasets.

Dataset	Image Source	# of Images (Train)	# of Images (Test)
CelebA	Real	$20k$	$10k$
	StarGAN	$20k$	$5k$
	AttGAN	$20k$	$10k$
	BEGAN	$20k$	$10k$
	ProGAN	$20k$	$10k$
	SNGAN	$20k$	$10k$
	MMDGAN	$20k$	$5k$
	CramerGAN	$20k$	$10k$
CelebA-HQ	Real	$20k$	$10k$
	ProGAN	$20k$	$10k$
	StyleGAN	$20k$	$5k$
	ResNet19	$20k$	$10k$
ImageNet	Real	$20k$	$10k$
	BigGAN	$20k$	$5k$
	S3GAN	$20k$	$10k$
	SNGAN	$15k$	$10k$
LSUN-Bedroom	Real	$20k$	$5k$
	ProGAN	$20k$	$10k$
	MMDGAN	$20k$	$5k$
	SNGAN	$20k$	$3k$
	CramerGAN	$20k$	$10k$
DTD	Real	-	$10k$
	ProGAN	$20k$	$5k$
FashionGen	Real	$20k$	$10k$
	DCGAN	$20k$	$5k$
Night	Real	$15k$	$5k$
	Pix2Pix	$15k$	$10k$
Shoes	Real	$20k$	$3k$
	Pix2Pix	$20k$	$10k$

Table 4.10: Initial labeled and discovery set for our online setup.

Dataset	**Labeled GANs**	**Discovery GANs**
CelebA	BEGAN, MMDGAN, CramerGAN, ProGAN	BEGAN, MMDGAN, CramerGAN, ProGAN, StarGAN, AttGAN, SNGAN
CelebA-HQ	ProGAN	-
ImageNet	SNGAN	-
LSUN Bedroom	ProGAN, MMDGAN, CramerGAN	ProGAN, MMDGAN, CramerGAN,SNGAN

real and fake image sources while grouping the different fake image sources together. This causes less discrimination between the fake image sources harming the clustering performance. We also provide a baseline (denoted by #) using our approach but adding JPEG and blur augmentations as used by [3]. We see that this degrades the performance compared to our original approach likely because these augmentations destroy valuable high frequency information used for discriminating between GAN sources. Since the baseline performance was lower in these evaluations we did not include them in the main paper.

4.5.4 Out-of-distribution detection

In this section, we provide more details on the WTA hash and also a comparison between cosine based distance and the WTA hashing based hamming distance for out-of-distribution detection. Additionally, we compare our approach with another popular out-of-distribution algorithm [5] and show that our algorithm performs well on their reported benchmarks. Finally, we analyze the effect of the percentile threshold used in our approach.

Table 4.11: Comparing the proposed approach with additional baselines from [2, 3]. * represents the pretrained features used for clustering while † denotes the features obtained from binary classification, the original task of [3]. We also provide a baseline (denoted by #) using our approach but with JPEG and blur augmentations as used by [3]. This does worse on both clustering metrics compared to our original approach.

# of clusters	Method	Avg. Purity	NMI
20	Wang *et al.*[3]*	0.1946	0.2042
	Yu *et al.*[2]*	0.4529	0.4543
	Wang *et al.*[3]†	0.3841	0.4434
500	Wang *et al.*[3]*	0.2929	0.2004
	Yu *et al.*[2]*	0.5947	0.3916
	Wang *et al.*[3]†	0.6082	0.4334
258	Ours$^{\#}$	0.7696	0.6249
266	Ours	0.8216	0.6552

4.5.5 WTA hash details

The WTA hashing algorithm proceeds as follows. Suppose a single feature vector $\boldsymbol{x}$ has a dimension d. We generate H different permutations $\boldsymbol{p}_i,\ i \in \{1, ..., H\}$ of indices $\{1, ..., d\}$ and then apply each of these permutations to $\boldsymbol{x}$ to get a set of vectors $\{\boldsymbol{x}'_i\}_{i=1}^{H}$. For each vector $\boldsymbol{x}'_i$, we take the first K elements, for a window size K, and obtain the index of the max element. The

Table 4.12: Comparison of our approach using WTA hash with ODIN [5]. The in-distribution dataset is CIFAR-100 which is used to train a DenseNet. We evaluate our method on the same metrics reported in [5]. The numbers reported are in the format of "ODIN/Ours". All values are in percentages. ↑ implies that the larger value is better while ↓ implies smaller value is better. We outperform ODIN on all OOD datasets and metrics excluding LSUN (crop).

Out-distribution dataset	FPR at 95% TPR ↓	Detection error ↓	AUROC ↑	AUPR In ↑	AUPR Out↑
Tiny-ImageNet (crop)	26.9/**18.8**	12.9/**10.2**	94.5/**96.4**	94.7/**96.6**	94.5/**96.3**
Tiny-ImageNet (resize)	57.0/**20.2**	22.7/**10.6**	85.5/**96.2**	86.0/**96.3**	84.8/**96.1**
LSUN (crop)	**18.6**/32.1	**9.7**/14.0	**96.6**/93.8	**96.8**/94.2	**96.5**/93.7
LSUN (resize)	58.0/**17.3**	22.3/**9.6**	86.0/**96.8**	87.1/**97.0**	84.8/**96.7**
iSUN	64.9/**28.3**	24.0/**12.6**	84.0/**94.8**	85.1/**95.2**	81.8/**94.6**
Gaussian	100.0/**0.0**	17.9/**0.1**	99.5/**100.0**	87.5/**100.0**	65.1/**99.8**
Uniform	100.0/**0.0**	38.0/**0.0**	40.5/**100.0**	60.5/**100.0**	40.9/**99.9**

set of these H indices (one for each permutation) yields a new vector $\boldsymbol{x}_H$. Note that $\boldsymbol{x}_H$ is a H dimensional vector with its elements taking integral values in $[0, K-1]$. The distance between two feature vectors is then defined as the hamming distance between their corresponding hashes.

4.5.5.1 Cosine based distance details

We compare the in-distribution, out-distribution and overall accuracy of our algorithm for the 12 seen classes (as described in Table 1 of the paper) using the WTA hash distance and a cosine-based distance. The results are shown in Table 4.13. As the number of samples in our in-distribution is roughly the same as number of samples in our out-distribution, we use standard accuracy as our metrics for comparison. In-distribution accuracy refers to the accuracy on all the samples in the discovery set which belong to the 12 seen classes while out-distribution corresponds to those belonging to the 8 unseen classes. Net accuracy is the overall accuracy on the full discovery set. We see that using the hash outperforms cosine based distance in terms of the net accuracy and in-distribution accuracy. It performs lower than the cosine-based distance in terms of the out-distribution accuracy but with only a small difference. This is because of an inherent tradeoff between in-distribution and out-distribution accuracy based on the percentile threshold.

4.5.5.2 Related works comparison for OOD

We now compare our approach with the popular out-of-distribution (OOD) approach called ODIN [5] on their benchmark. We show results using features extracted from DenseNet trained

Table 4.13: Comparison of our OOD step using WTA hash or cosine distance. We see that the WTA hash consistently outperforms the cosine-based distance at all 4 iterations of training even though it drops slightly on the out-distribution accuracy.

Iteration	In-distribution Accuracy (%)	Out-distribution Accuracy (%)	Net Accuracy (%)
1	86.02/**91.74**	**93.26**/89.35	89.33/**90.65**
2	83.49/**88.33**	**98.36**/97.63	90.22/**92.58**
3	81.14/**85.37**	**99.32**/98.34	89.45/**91.3**
4	79.11/**82.94**	99.10/**99.12**	88.27/**90.33**

Table 4.14: Comparison between using the WTA hash based hamming distance or the cosine based distance for computing the 1-NN graph during merge step. We analyze the performance directly at iteration 1 and also at the end of 4 iterations for both stages of merge and refine.

#iter.	Stage	Avg. Purity	NMI	% Samples Discovered	# of Sources Discovered	# of clusters
1	Merge	0.791/**0.793**	0.642/**0.646**	-	-	433/**383**
	Refine	0.776/**0.780**	**0.671**/0.666	74.70/**76.54**	4/**5**	**167**/180
4	Merge	0.820/**0.825**	0.635/**0.647**	-	-	618/**432**
	Refine	0.819/**0.823**	0.651/**0.655**	91.80/**94.76**	**8**/**8**	294/**266**

Table 4.15: Reducing number of clusters for K-Means (K) by almost half at each iteration. Average Purity and NMI does not change drastically compared to our default setup.

Iteration	K	Avg. Purity	NMI
1	500	0.695	**0.6756**
2	250	0.7877	0.6572
3	125	**0.8086**	0.6484
4	60	0.8056	0.6399
Ours (Default)		0.8216	0.6552

Table 4.16: We Naïvely recluster the test set after each training step and use them as pseudolabels for retraining. Compared to our original approach, a significant drop in Average Purity and NMI is observed.

Step	Avg. Purity		NMI	
	Reclustering	Ours	Reclustering	Ours
1	0.7858	0.7803	0.5402	0.6658
2	0.7803	0.8183	0.5383	0.6625
3	0.7597	0.8211	0.5289	0.6595
4	0.7303	0.8216	0.5112	0.6552

Table 4.17: Effect of the percentile threshold for OOD on the final performance. The default value for our experiments is 0.9. For all the thresholds, all sources were discovered.

β	Avg. Purity	NMI	# of Clusters	% Samples Discovered
0.7	0.7952	0.5842	449	0.9226
0.8	0.8087	0.6135	376	0.9234
0.9	0.8216	0.6552	266	0.9476
0.95	0.8268	0.6985	181	0.9358

on CIFAR-100. We evaluate on the various out-of-distribution datasets provided by the authors of [5] and report our results in Table 4.12. We see that we outperform their algorithm on almost all datasets except LSUN (crop). Additionally, our algorithm has very few hyperparameters which require careful tuning and does not require a validation set. This shows that our approach generalizes well to other dataset setups and can be used in general for the problem of out-of-distribution detection.

4.5.5.3 Threshold analysis

We evaluate the performance of our pipeline when varying the percentile threshold which was set by default to 0.9. We run our full pipeline iteratively for 4 iterations and report the numbers for 4 different values of 0.7, 0.8, 0.9, 0.95. The results are summarized in Table 4.17. Increasing the threshold has the expected result of decreased clusters as more samples in the discovery set are called in-distribution and are attributed to existing clusters. However, the resulting cluster metrics do not vary drastically in the neighbourhood of the default value of 0.9. This shows that our pipeline is fairly robust to the percentile threshold which also intuitively transfers across datasets. Our approach thus has a single easy to tune scalar which automatically varies the threshold for the different seen classes or clusters.

4.5.6 Clustering

We analyze the effect of the clustering, merge and refine stages of the pipeline, varying a number of hyperparameters and show our pipeline's robustness to these values with respect to the final performance.

4.5.7 Effect of different number of clusters and number of rounds of merge and refine

The k value chosen for K-Means changes the number of clusters that are passed on to the merge step. Additionally, we try performing multiple rounds of merge and refine within a single iteration, which decreases the number of clusters and improve purity. Table 4.19 summarizes the effect of changing k and the number of rounds of merge and refine, r, for the pipeline. We see that for a fixed r and different k's, even though the number of clusters changes in the clustering stage, the final number of clusters at the end of 4 iterations are similar and the performance on Average Purity and NMI does not vary drastically. However, when we fix k and vary r we see that final number of clusters show a visible drop and as expected NMI increases slightly while Average Purity decreases.

4.5.7.1 Cosine distance based merging

Instead of using the hamming distance between the WTA hashes of the feature vectors, we use a cosine based distance between feature vectors and compare the performance at the end of the first iteration and also at the end of 4 iterations. Table 4.14 compares cosine and WTA based

Figure 4.4: Some example clusters merged by our approach during the merge step.

distance at both points in the pipeline at the end of both merge and refine steps. We see that the WTA hash based distance marginally outperforms the cosine based distance at almost all points in the pipeline. It also discovers a higher percentage of samples with fewer clusters demonstrating its effectiveness in our pipeline.

4.5.7.2 Size threshold and SVM firing threshold

Next, we evaluate the effect of varying the size threshold, τ for discarding clusters at the end of refine step, as well as varying the SVM fire threshold, ϵ, which also controls the number of clusters (likely impure) being discarded. The results are summarized in Table 4.18. As expected increasing τ or ϵ decreases the number of clusters as more samples are discarded. This comes at the cost of fewer samples and GANs being discovered. However, the clustering metrics do not vary drastically showing that our pipeline is robust to these hyperparameter values and can generalize across varying datasets and sizes.

Table 4.18: Effect of varying the size threshold (τ) and SVM fire fire threshold (ϵ) on performance. The default setting corresponds to $\tau = 100$ and $\epsilon = 0.5$.

τ	ϵ	Avg. Purity	NMI	Sources Discovered	# of Clusters	% Samples Discovered
50	0.3	0.8178	0.6356	20/20	407	0.9695
	0.5	0.8212	0.6326	20/20	434	0.9729
	0.7	0.8223	0.6551	20/20	308	0.9549
100	0.3	0.8238	0.6451	20/20	293	0.9590
	0.5	0.8216	0.6552	20/20	266	0.9476
	0.7	0.823	0.6573	20/20	245	0.9237
200	0.3	0.8265	0.6731	20/20	166	0.8919
	0.5	0.8197	0.67	20/20	161	0.9015
	0.7	0.8233	0.686	19/20	138	0.8841
300	0.3	0.7849	0.7124	14/20	70	0.8643
	0.5	0.8009	0.6992	13/20	89	0.8340
	0.7	0.8055	0.7161	18/20	85	0.8638

4.5.7.3 Reclustering

We now evaluate the effect of removing the merge and refine steps from our pipeline. Our pipeline consists of the usual network training followed by clustering. For each iteration, we discard the old clusters and perform clustering again using K-Means on all the test set samples. We then use the new clusters as pseudo-labels and retrain our network for improving the features. It should be noted there is no OOD detection, merge or refine steps. This method is similar to ClusterFit [163] who also use cluster pseudo-labels for network training. We analyze the results in Table 4.16 by comparing with our original pipeline. We see that the performance drops by a significant margin showing that it is crucial to maintain existing clusters and iteratively merge and refine them while simultaneously improving the feature representations of the existing clusters.

Table 4.19: Effect of varying the number of clusters k for K-Means along with the number of times merge and refine (Additional Iters) is performed for each step. In the default setting Additional Iters is 0 as merge and refine are performed once per step while $k = 500$.

K	Additional Iters	Avg. Purity	NMI	Sources Discovered	% Samples Discovered	# of Clusters
100	0	0.7948	0.6722	20/20	0.9839	133
	1	0.7952	0.6938	20/20	0.9876	104
	2	0.7889	0.7016	19/20	0.9904	89
	3	0.7627	0.7255	19/20	0.9835	66
200	0	0.8086	0.657	20/20	0.9762	199
	1	0.8125	0.6798	20/20	0.9847	154
	2	0.7944	0.6981	19/20	0.9749	105
	3	0.7944	0.7085	20/20	0.9757	95
300	0	0.8142	0.652	20/20	0.9665	238
	1	0.8051	0.6696	20/20	0.9486	163
	2	0.802	0.6975	20/20	0.9602	121
	3	0.7669	0.7046	17/20	0.9569	92
400	0	0.8175	0.6472	20/20	0.9494	281
	1	0.814	0.6639	20/20	0.9637	196
	2	0.8105	0.6817	20/20	0.9687	158
	3	0.8005	0.7078	20/20	0.9679	113
500	0	0.8216	0.6552	20/20	0.9476	266
	1	0.8249	0.6736	20/20	0.9532	195
	2	0.8118	0.6887	20/20	0.9536	136
	3	0.7862	0.691	20/20	0.9485	114
600	0	0.8264	0.6535	20/20	0.9396	279
	1	0.8229	0.667	20/20	0.9545	212
	2	0.8176	0.6854	20/20	0.9470	153
	3	0.8039	0.6936	20/20	0.9459	121
700	0	0.8279	0.6572	20/20	0.9137	266
	1	0.8294	0.6837	20/20	0.9115	172
	2	0.8348	0.6887	20/20	0.9473	166
	3	0.8233	0.6923	20/20	0.9382	139

4.5.7.4 Effect of varying number of clusters

For most of our experiments we run the clustering using K-Means at a fixed value of k, which is used for all iterations. As the number of undiscovered samples reduce as number of iterations increases, we evaluate our pipeline's performance by decreasing k after each iteration. We thus, approximately halve the value of k after each iteration. The results are reported in Table 4.15. We see that the performance does not change drastically compared to the default setup which shows that our network does not heavily rely on the number of clusters used during K-Means.

4.5.7.5 Qualitative Analysis

We now qualitatively show the effect of our merge step for a few clusters. The merge step of our approach merges clusters belonging to the same class but are actually fragmented due to overclustering from K-Means. We visualize two such clusters in Fig. 4.4. As we showed in the main paper, our clusters focus on the GAN source rather than image semantics and the merge step successfully combines clusters having the same majority GAN source.

4.5.8 Network Training

4.5.8.1 Effect of faster training

By default, we retrain all feature extractor weights every iteration. To reduce the cost of full network retraining, we analyze finetuning only the final residual block of the ResNet-50 backbone along with the subsequent fully connected layers of the feature extractor. We also analyze using a

Table 4.20: Results on our setup with slight variations in our training.

Experiment	# of Clusters	Avg. Purity	NMI	# Sources Disc.
Ours (Original)	209	0.861	0.724	20/20
Ours (w/o merge)	257	0.841	0.712	19/20
Ours (Freeze)	229	0.850	0.691	19/20
MobileNet	70	0.846	0.773	15/20

lighter network such as MobileNet [198] for our network training and compare it with our original setup. Additionally, we try to see the effect of removing the merge step from the pipeline The results are summarized in Table 4.20. We see that there is a small drop in network performance in terms of both Average Purity and NMI and it also fails to discover a single unseen source. Constraining the network is likely to have restricted the network's capability of improving the discovery set features although it doesn't have a significant impact. On the other hand, MobileNet obtains a higher NMI because of much fewer clusters, albeit at the cost of not discovering most of the unseen sources. This shows that very light networks are not as effective in obtaining discriminative representations for discovering new sources. Also the performance without merge is sub-par to our original approach which shows the importance of performing merge step to group similar clusters together after over-clustering.

4.5.8.2 Effect of image size

By default, we resize and center crop all images to 256×256 in most of our experiments. We compare results when image sizes are varied from 64 to 256 for network training. From Table 4.21, we see that increasing image size shows a marked improvement in all metrics. Therefore, we hypothesize that model fingerprints are likely more detectable and distinguishable when the image is resized to a higher resolution. However, this comes at the cost of increasing network

Table 4.21: Results on our setup with varying image sizes.

Image Size	# of Clusters	Avg. Purity	NMI	# Sources Disc.
64	169	0.655	0.579	19/20
128	266	0.822	0.673	20/20
256	209	0.861	0.724	20/20

Table 4.22: Results on our setup with variations in training.

Experiment	# of Clusters	Avg. Purity	NMI	# Sources Disc.
Ours (Original)	209	0.861	0.724	20/20
Ours + Unseen seeds	216	0.842	0.702	21/22

training times and memory requirements (quadratically) which is infeasible in an online setup or for very large scale datasets.

4.5.9 Multiple seed sources

[2] showed that training generators with different random seeds generate different distinguishable fingerprints in their images. We analyze whether we can discover new separate sources when a single generator architecture is trained on the same dataset but with different seeds. Table 4.22 shows results comparing this setup with our original setup. We add 2 different seeds for ProGAN for both CelebA and LSUN-Bedroom providing 2 new sources. Note that only a single seed of ProGAN trained on LSUN-Bedroom is present in the labeled set while the other 3 sources are unseen. The remaining classes are same as our original setup as described in Table 1 of the main paper. We see that there is only a small drop in Average Purity and NMI although it fails to discover a single unseen source.

Part II: Data and Supervision Strategies in Generation

Chapter 5: GRIT: GAN Residuals for Paired Image-to-Image Translation

Figure 5.1: We decouple the optimization of reconstruction and adversarial losses by synthesizing an image as a combination of its reconstruction (low-frequency) and *GAN residual* (high-frequency) components. The GAN residual adds realistic fine details while avoiding the pixel-wise penalty imposed by reconstruction losses.

Generative Adversarial Networks (GANs) have had a revolutionary impact on generative modeling and image synthesis. In their unconditional setting [8, 177, 199], GANs map a source distribution, typically a unit Gaussian, to a target distribution (*e.g.*, real images). At inference time, random images can be synthesized by sampling latent codes from the source distribution and passing them through a generator network. To provide user control over the synthesis process, Isola *et al*. [192] proposed a GAN-based Image-to-Image (I2I) translation framework, which conditions the synthesis process on an input image that describes certain attributes of the target output. Therefore, I2I translation learns to map images from a source domain A to a target domain B (*e.g.*, semantic maps $\rightarrow$ scenes or sketches $\rightarrow$ photo-realistic images). I2I translation has since been utilized for many problems in computer vision and graphics, such as inpainting [200],

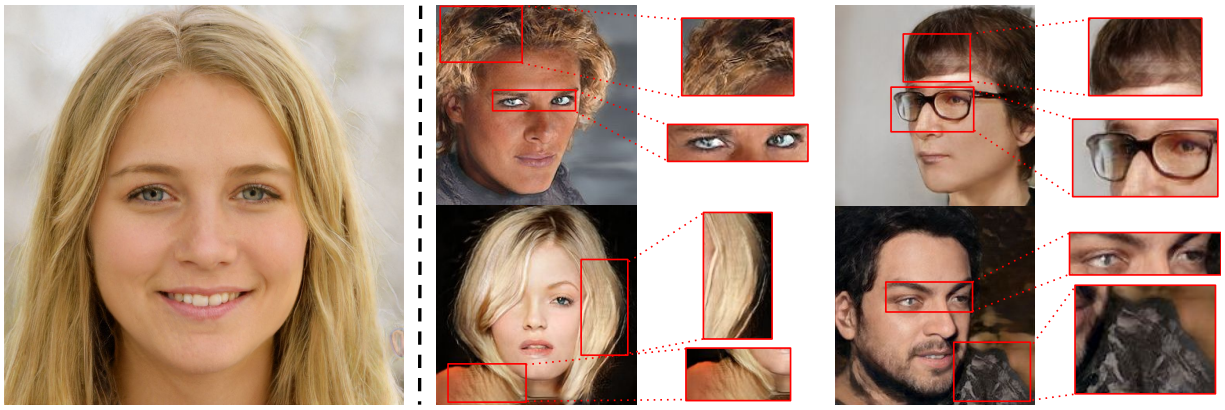


Figure 5.2: Comparing image realism between unconditional GANs and I2I translation. Left: Sample output from *StyleGAN* [8] at 1024×1024 resolution. Right: I2I translation outputs from *GauGAN* [9] at 256×256 resolution. Even at a lower resolution, I2I shows more noticeable artifacts compared to unconditional GANs.

colorization [201, 202], super-resolution [203], image de-noising [204], rendering [205, 206, 207], and many more [208, 209, 210].

While unconditional GANs [211, 212] and class-conditional GANs [183] have reached unprecedented visual quality, I2I translation lags behind in quality and realism. This is despite the fact that it has more inputs and better supervision during training. For example, Figure 5.2 contrasts *StyleGAN* [8] (unconditional) and *GauGAN* [9] (I2I) which both came out around the same time and from the same institution. Yet, there is a clear realism gap in favor of unconditional GANs.

We are motivated by this realism gap between unconditional GANs and I2I translation. We investigate the cause of this performance gap and trace it back to the difference in the training objective between those two tasks. In unconditional GANs, the generated output is supervised only by an adversarial loss $\mathcal{L}_{\text{adv}}$, where a critic/discriminator network learns to score how realistic the output looks. On the other hand, I2I translation relies on cyclic [213] and cross-cyclic [214, 215] reconstruction losses between the generated output and available ground truth images.

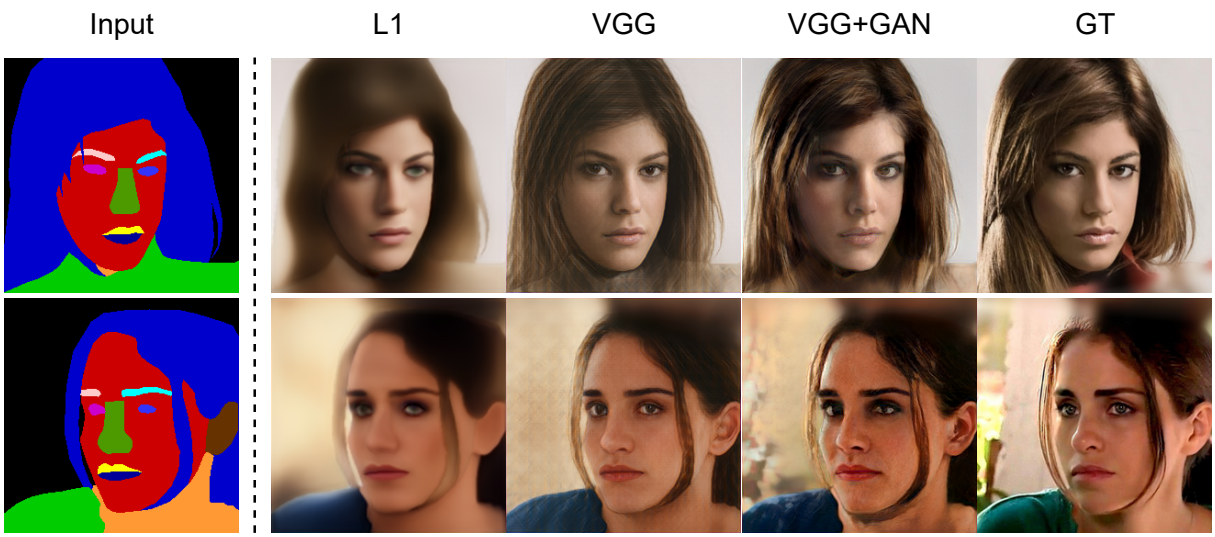


Figure 5.3: Comparison between different I2I training objectives: Left is the input semantic layout. The following columns show the output of networks trained with an $L1$ loss, VGG-based perceptual loss, perceptual+adversarial (GAN) losses respectively. Last column shows the corresponding ground truth image.

Thus the training objective of I2I translation optimizes a weighted combination of both an adversarial loss, $\mathcal{L}_{\text{adv}}$, and a reconstruction loss, $\mathcal{L}_{\text{rec}}$. Reconstruction losses enforce a form of pixel-wise matching between the ground-truth image I^B and the output reconstruction $\hat{I}^B$. This provides a strong supervisory signal which speeds up convergence significantly when compared to unconditional GANs. However, we show that reconstruction losses are at odds with adversarial losses, which does not lead to a sound optimization objective and causes visual artifacts in I2I outputs.

Figure 5.3 shows the effect of optimizing different objectives for I2I translation. Optimizing an $L1$ reconstruction alone leads to very blurry outputs. While using a VGG-based perceptual loss [216] achieves much better results, the output is not sharp and contains clear grid artifacts. Adding an adversarial loss brings the output closer to the distribution of real images, but, in many cases, artifacts can be spotted (*e.g.*, around the hair, teeth and eyes). We hypothesize that

directly optimizing a reconstruction loss on the output ignores a type of multi-modality in image synthesis, which leads to visible artifacts. To motivate our hypothesis, Figure 5.4 shows how GANs improves realism by simulating fine details found in real images, like local variations or noise patterns found in the texture of real materials. There are infinite realizations of such noise patterns that add realism to the output (*e.g.*, skin freckles, pores and wrinkles, and linings of hair strands). However, applying a reconstruction loss ($\mathcal{L}_{\text{rec}}$) in traditional I2I frameworks penalizes all these local variations and promotes a uni-modal solution where the generated image matches the ground truth down to the pixel level. This leads to smoothed outputs and other noticeable artifacts that do not show in the unconditional GAN setup where no reconstruction loss is applied.

In this work, we address this problem and propose a modified architecture and training objective that relaxes the role of reconstruction losses to act as regularizers instead of doing all the heavy lifting which is common in current I2I translation frameworks. Our formulation decouples the optimization of adversarial and reconstruction losses. This enables our network to hallucinate local variations to add realism to the output while avoiding being penalized by reconstruction losses. Although we investigate our proposal in a paired I2I setting, the idea can be extended to unpaired I2I.

We summarize our contributions as follow:

- We study the realism gap between unconditional GANs and paired I2I translation, and shed light on an important multi-modal aspect of image synthesis that we denote as *local spatial variations*, which is overlooked and rather penalized in traditional I2I translation formulation.
- Through the proposed approach, we use GAN Residuals for Image-to-Image Translation (GRIT),

Figure 5.4: Examples of multi-modal outputs (generated by our method) with local stochastic variations that add realism and satisfy the GAN objective. Applying reconstruction losses in traditional I2I frameworks ignores this type of multi-modality and penalizes such variations, which misleads I2I training.

and take the first step towards addressing the multi-modal nature of local spatial variations in I2I translation. We utilize a modified architecture and training objective that models and encourages such multi-modality.

- We provide quantitative and visual evidence on the effectiveness of modeling local spatial variations in paired I2I translation, and show that our proposed method improves upon strong baselines.

5.1 Related work

Since the onset of the GAN era with the seminal work of Goodfellow *et al*. [199] there have been multiple works [177, 178, 181, 183] to improve the synthesis quality and resolution of images. Karras *et al*. [211] improved the quality of [181] by introducing better normalization

and regularization and in the process reduced image artifacts and made inversion easier. [212] went a step further to reduce aliasing and also proposed an approach which made representations equivariant to rotation and translation. While these works try to learn the manifold of training data to generate samples on it, there is another synthesis task of Image-to-Image (I2I) translation which has been vastly explored and involves translating images from one domain to another. I2I translation can be broadly grouped into two regimes based on the type of training data, unpaired data or paired data.

Unpaired Image-to-Image Translation utilizes unpaired training data which does not have pixel level correspondence between the domains. CycleGAN [217], DualGAN [218] and DiscoGAN [219] proposed one of the first and most commonly used approaches for this involving a cyclic loss to impose consistency between forward and backward translation for the same image. UNIT [220] and SCAN [221] also utilize the cyclic loss but introduce a shared latent space and multistage coarse to fine training respectively. TransGaGa [222] extends the cyclic loss to large domain gaps by disentangling features into appearance and geometry latent space. On the unidirectional (non-cyclic loss) end, approaches like DistanceGAN [223] train by maintaining distance between pairs of samples. GcGAN [224] enforces constraints on geometric transformations preserving image semantics. CUT [225] proposes a multi-layer patch based contrastive learning approach while MUNIT [214] and DRIT [226] disentangle representations into style and content. [227] also uses disentanglement but into texture and structure. More recently MSPC [228] proposes a maximum spatial perturbation consistency based regularization.

Paired Image-to-Image Translation uses paired data making it possible to enforce pixel level correspondence. One of the first works in this direction, Pix2Pix [192] proposed an $L1$ reconstruction loss with a patch discriminator. Later, Pix2PixHD [229] improved it with higher reso-

lution generation using coarse to fine generator and multi-scale discriminator. DNI [230] looks at decoupling reconstruction and GAN losses but they end up training two separate models with differing objectives and then interpolating between them by performing a weighted summation to get a balance between both tasks. They also assume some level of correlation between the parameters of the two networks for this to work and lack quantitative evidence for the efficacy of their approach. SPADE [9] proposed spatially-adaptive normalization layer as vanilla normalization washes away semantic information. SEAN [231] further proposed semantic region-adaptive normalization layer to control style of each semantic region separately. CoCosNet [232] jointly learns the cross domain correspondence and image translation, where both tasks facilitate each other and thus can be learned with weak supervision. Later CoCosNetv2 [233] mitigated the quadratic complexity issue in CoCosNet and enabled high-resolution correspondence using PatchMatch [234]. Recently DINO [10] proposed an energy based cyclic framework to utilize the conditional input. While MoNCE [11], presents a re-weighted patch based constrastive learning framework. Unlike these works in our approach we disentangle the reconstruction and adversarial (GAN) loss. Additionally, we also propose architecture modifications which enable us to perform this disentanglement by separating the reconstruction supervised output and the residual and in the process make better use of per-pixel spatial noise to learn more realistic and diverse I2I translations.

5.2 Approach

The goal of reconstruction losses is to guide a generated output $\hat{I}^B$ to resemble a target ground truth image I^B. While this is a desired behavior for I2I translation, a negative side ef-

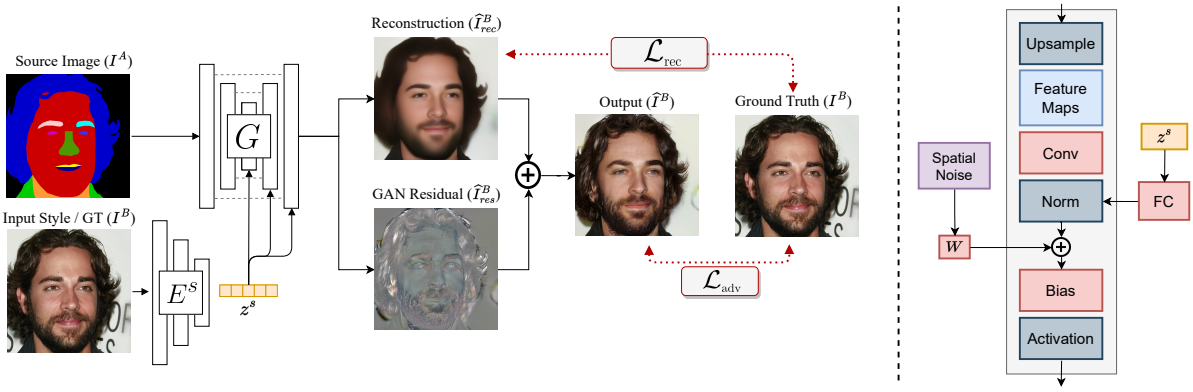


Figure 5.5: **Left:** : Overview of GRIT. Our network generates the output as the composition of a reconstruction component $\hat{I}^B_{\text{rec}}$ and a *GAN-residual* component $\hat{I}^B_{\text{res}}$. An $L1$ reconstruction loss is applied only to the reconstruction component, while the GAN residual is supervised only through an adversarial loss $\mathcal{L}_{\text{adv}}$. **Right:** The generator's upsampling block. We feed the encoded style latent z^s through *AdaIN* layers, and also add random spatial noise maps controlled by learnable weights W to the feature maps.

fect is that a reconstruction loss will also penalize high frequency deviations between $\hat{I}^B$ and I^B. Therefore, this formulation ignores the multi-modal nature of synthesizing fine-grain texture patterns, where there are infinitely many realizations of local high-frequency details (*e.g.*, skin texture or the location of hair strands as shown in Figure 5.4). Penalizing such local variations and promoting a uni-modal solution thus causes artifacts and contributes to the realism gap between unconditional GANs and I2I translation.

Through our approach GRIT, we make a first step towards addressing this overlooked multi-modal aspect of image synthesis, and propose to decouple the optimization of reconstruction and adversarial losses.

We present our formulation in Section 5.2.1 and associated changes to the loss function in Section 5.2.2. Section 5.2.3 discusses how to explicitly model multi-modal local variations for paired I2I translation.

5.2.1 Formulation

We propose to generate an image $\hat{I}^B$ as the composition of two components: a reconstruction component $\hat{I}^B_{\text{rec}}$, and an adversarial *GAN-residual* component $\hat{I}^B_{\text{res}}$. During training, this decoupling of $\hat{I}^B_{\text{rec}}$ and $\hat{I}^B_{\text{res}}$ allows the reconstruction component $\hat{I}^B_{\text{rec}}$ to focus on reconstructing low-frequency details of the target real image I^B, while the *GAN-residual* component $\hat{I}^B_{\text{res}}$ hallucinates high-frequency details that add realism to the synthesized image $\hat{I}^B$. The final output is generated as:

$$\hat{I}^B = \mathcal{C}(\hat{I}^B_{\text{rec}}, \hat{I}^B_{\text{res}}); \qquad \hat{I}^B_{\text{rec}}, \hat{I}^B_{\text{res}} = G(I^A, z^s) \tag{5.1}$$

where $G(.,.)$ is a generator network that maps an input image I^A along with a style latent code z^s to its low- and high-frequency components $\hat{I}^B_{\text{rec}}, \hat{I}^B_{\text{res}}$, and $\mathcal{C}(.,.)$ is a composition operator that combines both output components. We implement $\mathcal{C}$ as a simple addition. We also investigated implementing it as a small CNN head that fuses $\hat{I}^B_{\text{rec}}, \hat{I}^B_{\text{res}}$ but found that simply adding the two images works better in our case and is more stable to train.

Figure 5.5 gives an overview of our architecture. We implement the generator network as a U-Net [235] architecture that consists of a content encoder E^C and a decoder D. The decoder network outputs both $\hat{I}^B_{\text{rec}}, \hat{I}^B_{\text{res}}$. We further discuss the decoder architecture in Section 5.2.3.

To model style multi-modality, we follow the literature [9, 213, 236] by utilizing a style encoder E^S that learns to capture the style of an input image into a latent style code z^s, which is fed to the generator G via *AdaIN* layers [237] to specify the style of the output $\hat{I}^B$. Next, we discuss our modification to the loss function to encourage the decomposition of $\hat{I}^B$ into its reconstruction and GAN-residual components.

5.2.2 Loss function

Standard loss function of GAN-based I2I translation networks consists of a weighted sum of a pixel-wise reconstruction loss $\mathcal{L}_{\text{rec}}$ and a discriminator-based adversarial loss $\mathcal{L}_{\text{adv}}$.

Minimizing this loss does not take into account possible local variations, as it promotes pixel-wise matching between the output $\hat{I}^B$ and the ground truth I^B, and thus only accepts one solution and penalizes any high-frequency variations. To allow local variations, we aim to only reconstruct the low-frequency components of a ground truth image I^B, where low-frequency components capture the *general* content and style of the target output. On the other hand, we want the generator to have the freedom to add fine-grain details, represented by high-frequency components, making the output photo-realistic.

We achieve this by modifying the loss to apply the reconstruction loss $\mathcal{L}_{\text{rec}}$ only to the reconstruction component $\hat{I}^B_{\text{rec}}$, while the adversarial loss $\mathcal{L}_{\text{adv}}$ is applied to the final output $\hat{I}^B = \mathcal{C}(\hat{I}^B_{\text{rec}}, \hat{I}^B_{\text{res}})$. Thus, our modified training objective is given by:

$$\min \mathcal{L}(I^B, \hat{I}^B, \hat{I}^B_{\text{rec}}) = \mathcal{L}_{\text{adv}}(\hat{I}^B, I^B) + \lambda_{\text{rec}}\mathcal{L}_{\text{rec}}(\hat{I}^B_{\text{rec}}, I^B) \tag{5.2}$$

With such modification, the reconstruction loss $\mathcal{L}_{\text{rec}}$ does not backpropagate into the GAN-residual component $\hat{I}^B_{\text{res}}$, and $\hat{I}^B_{\text{res}}$ therefore has the freedom to hallucinate high-frequency details that add realism to generated images without being constrained to match pixel-level details of ground truth images at training time. While the proposed loss function (Eqn. 5.2) allows high-frequency deviations between I^B and $\hat{I}^B$, this by itself does not encourage multi-modal synthesis of local texture and other high-frequency details. In the next section, we discuss how to explicitly

model the local-variations multi-modality into our network.

5.2.3 Multi-modal outputs

At training time, I2I translation networks peek at the target ground truth image I^B and encodes it into a flattened style latent code z^s. However, due to the lossy nature of such compression, it is impossible for the decoder to recover pixel-level spatial information (*e.g.*, location of hair strands) to reconstruct I^B. Driven by the adversarial loss, the decoder hallicinates spatial patterns to bring synthesized images closer to the manifold of real images. This requires the decoder to devise a way to generate spatially-varying pseudo-random numbers from the input flattened latent. This challenge was first raised in StyleGAN [8], where they showed that this is inefficient and consumes much of the network capacity. To address this limitation, Karras *et al.* [8] proposed to add per-pixel noise maps within each upsampling block in the decoder to encourage synthesizing local variations of spatial patterns.

The use of spatial noise maps however did not transfer to the I2I translation literature. This is because, unlike unconditional GANs, the application of reconstruction losses counteracts the added spatial noise by suppressing it, leading to a uni-modal output. On the other hand, decomposing the synthesis into its reconstruction $\hat{I}^B_{\text{rec}}$ and GAN-residual $\hat{I}^B_{\text{res}}$ components allows for naturally adapting spatial noise maps to I2I translation by modeling local stochastic variations in the GAN-residual component. This bridges the gap between unconditional GANs and I2I translation since $\hat{I}^B_{\text{res}}$ is not affected by the reconstruction loss, and can therefore fully utilize the added spatial noise. Adding spatial noise maps models the local variation multi-modality, and enables generating multi-modal output for the same target image by sampling random noise

maps.

Figure 5.6: Qualitative comparison on CelebAMask-HQ dataset with DINO [10], MoNCE [11], Pix2PixHD [12] and GauGAN/SPADE [9].

5.3 Experimental evaluation

Dataset. We perform our main evaluation on the CelebAMask-HQ dataset [238]. The dataset contains $30,000$ high resolution face images along with their corresponding segmentation masks which contain 19 semantic labels and are at a 512×512 resolution. We use the standard train and test splits provided by Liu *et al.* [173]. We also show results on Edges2Handbags [239] which contains 137K Amazon Handbag images and edge maps. All images and edge maps are at 256×256 resolution. Unless stated otherwise, all experiments and analysis are performed on

Figure 5.7: Qualitative comparison on Edges2Handbags dataset with Pix2PixHD [12] and SPADE [9].

the CelebAMask-HQ dataset [238].

Baselines. We compare our method with the following approaches: Pix2PixHD [229], SPADE (also called GauGAN) [9], DINO [10] and MoNCE [11]. We train Pix2PixHD [229] and SPADE [9] using their official released code. For Pix2PixHD, we enable the option to train a semantic-specific style encoder, which computes separate style codes per semantic label. We use the outputs provided by the authors for DINO [10], and use the released pre-trained model of MoNCE [11] and follow the authors' instructions to generate test results on the CelebAMask-HQ dataset. Since MoNCE and DINO are trained at 256×256, we train our method as well as Pix2PixHD and

Table 5.1: Comparison with baselines at 256×256 resolution.

Method	L1 ↓	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓
Pix2PixHD [12]	24.78	17.53	0.515	0.256	45.97
SPADE [9]	28.69	16.21	0.485	0.283	26.06
DINO [10]	51.84	12.13	0.401	0.369	37.24
MoNCE [11]	64.26	10.32	0.357	0.380	34.27
Ours	**18.34**	**19.54**	**0.531**	**0.245**	**17.04**

Table 5.2: Comparison on Edges2Handbags at 256×256 image resolution.

Method	L1 ↓	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓
Pix2PixHD [12]	19.28	18.05	0.70	0.20	59.53
SPADE [9]	21.91	16.71	0.66	0.25	75.49
Ours	**12.61**	**20.68**	**0.73**	**0.18**	**57.79**

SPADE at 256 resolution for fair comparison. Additionally, we also train our method, Pix2PixHD and SPADE at 512×512 resolution to evaluate and compare results at high resolution. **Metrics.** We evaluate using the following metrics:

- Standard reconstruction metrics such as $L1$, Peak Signal to Noise Ratio (PSNR), and structural similarity (SSIM) [240] between the output and ground truth.
- LPIPS [241] which measures the perceptual similarity between the output and ground truth using AlexNet features.
- Frechet-Inception Distance (FID) [242] which is used to measure the perceptual quality and realism of the output.

5.3.1 Quantitative Comparison

We provide quantitative comparison with the baselines in Table 5.1 and 5.2 for the CelebAMask-HQ and Edges2handbags datasets respectively which are commonly used in paired I2I literature. For CelebAMask-HQ dataset, we observe that Pix2PixHD performs much better than SPADE on reconstruction metrics like L1, PSNR, SSIM and LPIPS for both the datasets. This is because Pix2PixHD uses a powerful semantic-specific style encoder that encodes a separate style code per each semantic label and is therefore able to match the ground truth style more accurately. On the other hand, SPADE uses a VAE-based encoder [243] which adds robustness to noise in the style latent space, but at the expense of faithful reconstruction of the ground truth style. SPADE however maintains good realism, and thus performs much better than Pix2PixHD on the FID metric.

While DINO [10] and MoNCE [11] are more recent baselines, we observe they fall short in comparison with Pix2PixHD and SPADE.

Finally, our decoupled optimization of reconstruction and adversarial losses achieves better reconstruction error, as well as better realism (FID) score compared to the baselines. Our reconstruction component $\hat{I}^B_{\text{rec}}$ focuses on reconstructing low-frequency details to match the general color and structure of the ground truth. Matching low-frequency components has a direct impact on reconstruction metrics, especially L1 and PSNR. Additionally, unlike the baselines, our GAN residual component $\hat{I}^B_{\text{res}}$ is not constrained by reconstruction losses. And so, it has the freedom to add high-frequency details that improves the output realism, which leads to a better FID score. Similar trends hold for the Edges2Handbags dataset with Pix2PixHD performing better than SPADE on all metrics including FID. This is because SPADE is designed for dense spatial

inputs, *e.g.*semantic maps, not sparse edge maps as in the case of Edges2Handbags.

While many baselines are trained at a 256×256 resolution, we also inspect our performance at a higher resolution of 512×512 on the CelebAMask-HQ dataset. To provide comparative evaluation at this resolution, we choose the Pix2PixHD and SPADE methods which are the top performing methods at 256×256 resolution, and retrain them at a 512 resolution. Table 5.3 shows that the proposed method consistently shows similar trends of improvement over the baselines across all metrics.

5.3.2 Qualitative evaluation

In this section we qualitatively analyze and compare synthesized results between our method and the baselines. We also look at various aspects of our approach through visual results to understand different components better.

5.3.2.1 Comparison with Baselines

Figure 5.6 shows qualitative comparison with the baselines on the CelebAMask-HQ dataset. Our method clearly improves over the baselines in terms of both realism, as well as matching the ground truth style. We note that our results could show some style deviations from the ground truth style (*e.g.*, lip color in the second row), we are still noticeably better than the baselines. We observe that the added GAN residual can sometimes cause such slight deviation from the reconstructed color, since it is not constrained by the reconstruction loss.

While DINO captures the structure well, it loses out on realism and on matching colors and textures to the ground truth image. MoNCE shows more details due its patch based nature during

Table 5.3: Comparison with baselines at resolution of 512×512.

Method	L1 ↓	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓
Pix2PixHD [12]	24.80	17.40	0.534	0.354	24.79
SPADE [9]	31.44	15.53	0.490	0.389	20.80
Ours	**19.02**	**19.36**	**0.555**	**0.333**	**16.91**

Table 5.4: Ablation of different components of our approach.

Method	L1 ↓	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓
U-Net [235]	21.68	18.25	0.504	0.222	20.24
+ spatial noise	20.93	18.55	0.520	**0.219**	21.41
+ GAN residuals (**ours**)	**18.34**	**19.54**	**0.531**	0.245	**17.04**

training, but again is not able to faithfully capture the style and structure well. Pix2PixHD and SPADE both generate reasonable results, but we observe that SPADE results look more realistic, although not faithfully matching the ground truth style. Our output on the other hand generates high quality and realistic samples while making sensible light deviations which capture the true nature of real world data.

We show results on Edges2Handbags dataset in Figure 5.7 where we observe similar trends as the Pix2PixHD output looks better than SPADE and has fewer artifacts while ours looks the most faithful. As can be seen from the figure our method generates the color, texture and structure better compared to the other approaches.

5.3.2.2 Standard Deviation of Spatial Noise

In our approach we utilize spatial noise by adding it to the feature maps at each upsampling block. This along with the decoupled objective lets the network learn to generate variations of lo-

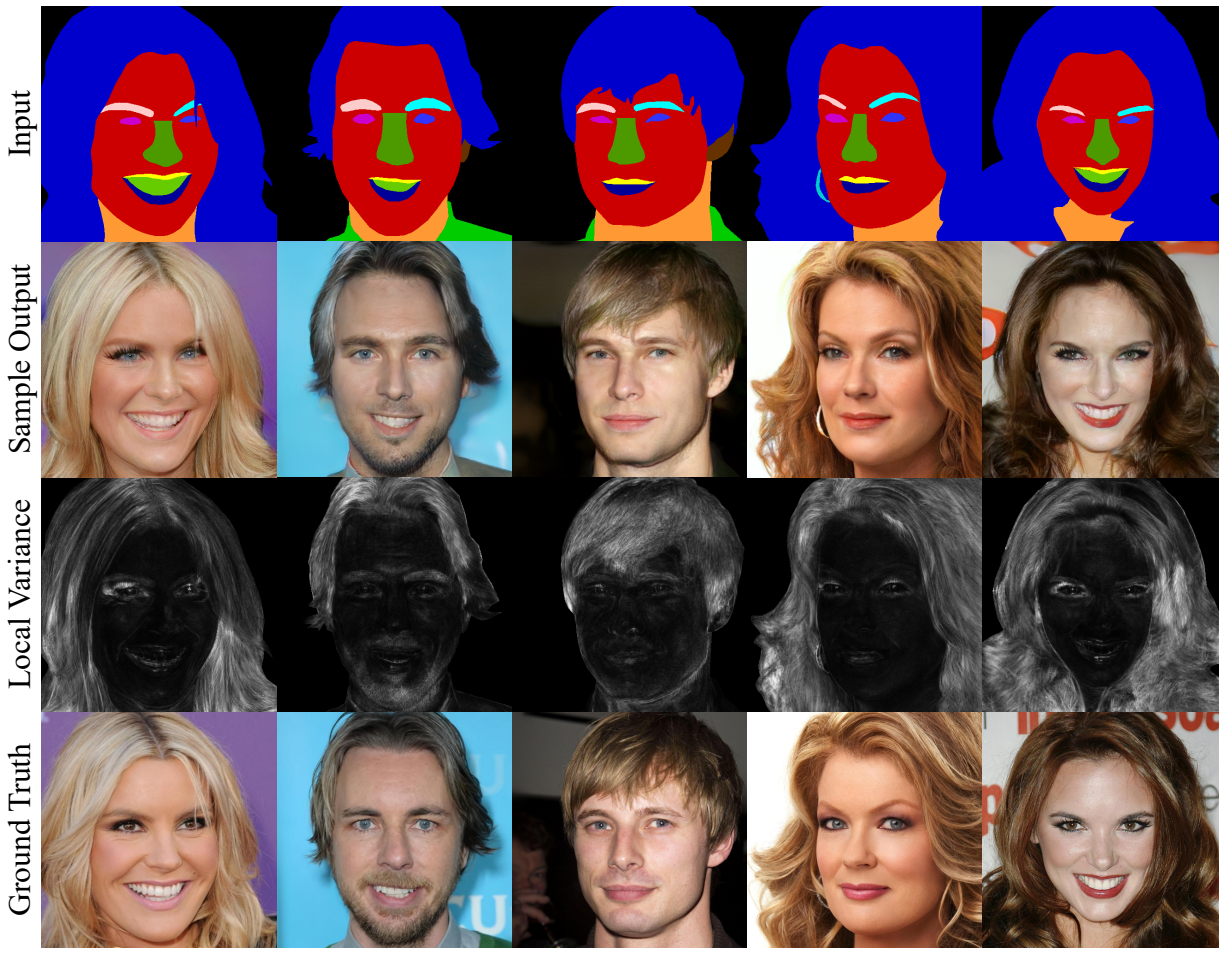


Figure 5.8: Examples of local stochastic variations. Top to bottom rows represent the input image, one sample output, standard deviation of each pixel over 20 different outputs for the same sample, and ground truth image respectively.

cal information which preserves the structure and content but introduces diversity in the generated samples. Here we analyze the variations generated for multiple subjects over 20 different spatial noise samples for each of them to understand the stochasticity better. Figure 5.8 shows pixel-wise standard deviation over the different translation results generated by varying the spatial noise on CelebAMask-HQ dataset. It can be see that highest deviation occurs in regions corresponding to hair, around eyes, lips and nose. These regions can be considered high-frequency locations as they usually contain multiple edges and have the most variations. By visualizing the standard deviation we are able to verify that the network is able to understand and model these regions

better and generate sensible variations.

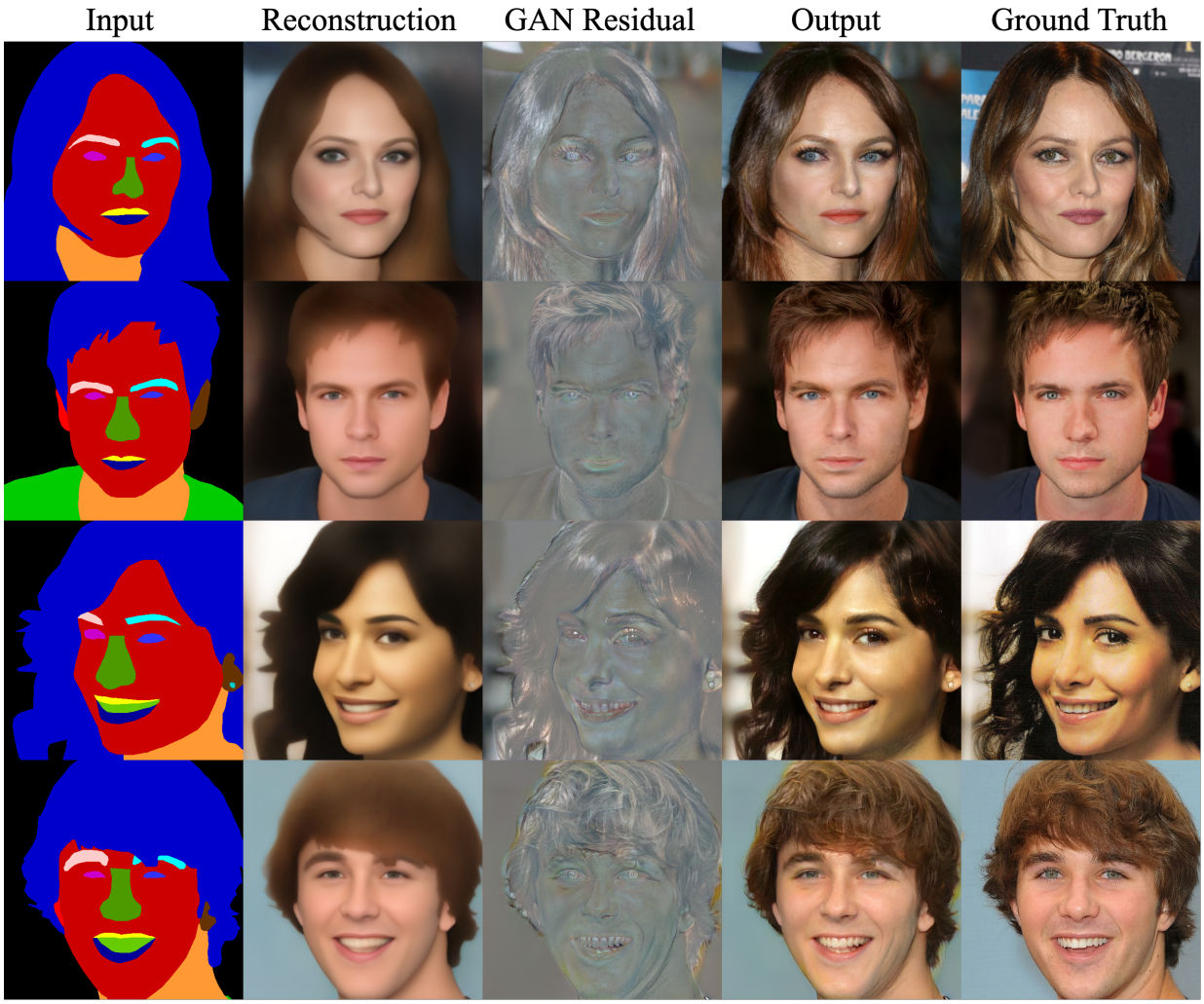


Figure 5.9: Examples of the different outputs of our method along with the input label map and ground truth image.

5.3.3 Ablation

We perform ablation our approach to show the effect of sequentially introducing each component using the CelebAMask-HQ dataset. These results are highlighted in Table 5.4. The first row shows the performance of a vanilla I2I framework which utilizes a U-Net [235] based generator with a reconstruction and GAN loss on the output. The second row corresponds to introducing

spatial noise which lets the model learn to generate local variations. It should be noted that introducing spatial noise on its own does not realize its full potential as the reconstruction loss can fight back and teach the network to ignore it in order to improve on the pixel-wise reconstruction loss. This is where the role of residuals comes in, which can be seen in the third row. Introducing the GAN residuals along with the spatial noise gives a considerable boost in performance, as while the reconstruction losses supervise the reconstructed output, the GAN loss supervises the combined output and lets the network learn residuals which can better capture details in the image.

5.3.4 Understanding the GAN Residuals

While Figure 5.8 shows examples of the different outputs generated by our network, namely, the reconstructed and residual images followed by the final combined image. Here we try to understand what kind of information these images hold. As can be seen from Figure 5.9 the reconstructed image encodes most of the structure and content of the image. It looks like a low-frequency and smooth image while the the residual seems to contain a lot of high-frequency information around the hair, eyes, beard, lips etc. where a lot of edges and variation occur. As can be seen in the combined output, adding these two gives a realistic image which resembles the ground truth.

We refer to the GAN residuals being high-frequency by capturing local variations. Here we verify this by computing the frequency spectrum of the images in a similar manner as Schwarz *et al*. [244] who use azimuthal averaging over the spectrum in normalized polar coordinates. In Figure 5.10 we show the average over all the synthesized outputs corresponding to the test set

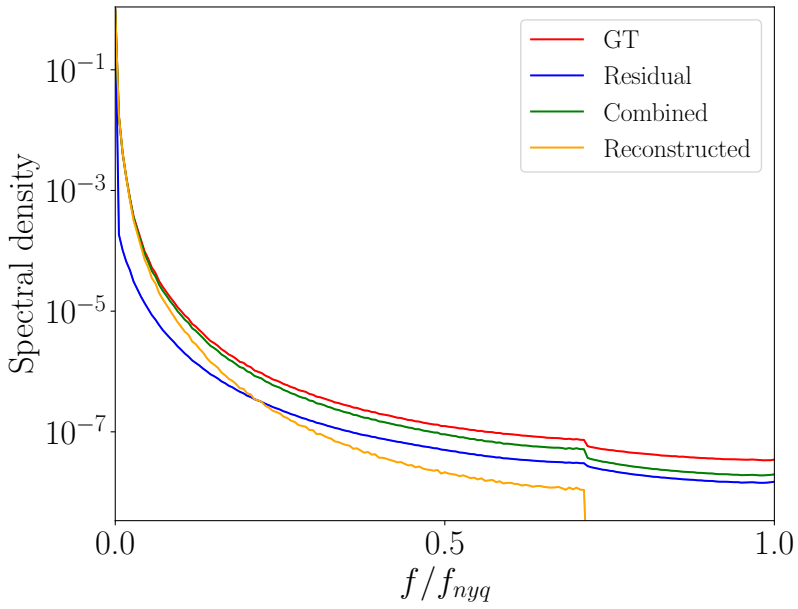


Figure 5.10: We visualize the frequency spectrum and highlight that the reconstructed image contains higher magnitude of low-frequency information while the residual captures the high-frequency more. By combining these, the resulting image has a spectrum closer to the ground truth image. The y-axis denotes the spectral density which is measures the magnitude of a particular frequency while the x-axis corresponds to the frequency relative to the maximum frequency corresponding to f_{nyq}.

for CelebAMask-HQ dataset. It can be seen how the the reconstruction (orange) encodes higher magnitude for the low-frequencies with a complete cutoff at mid-to-late frequencies. On the other hand the residual (blue), encodes more of the high-frequency information. Combining both of them (green) is much closer to the frequency spectrum of the ground truth (red).

5.4 Discussion

We propose a novel approach for paired image-to-Image translation by highlighting the disconnect between the reconstruction and adversarial losses which are at odds with each other. Based on this insight we decouple the reconstruction and adversarial loss in the proposed approach which enable it to have the freedom to learn local variations better and generate more

realistic translations. Through both quantitative and qualitative results we highlight the efficacy of the proposed approach and achieve state-of-the-art performance on paired I2I task on both CelebAMask-HQ and Edges2Handbags datasets. We show results and compare at both 256×256 and 512×512 resolutions which shows that the proposed method can generate higher resolution images too. We also analyze the diversity in image synthesis that our method introduces using the spatial noise and highlight its relation to high-frequency. Further, we analyze the residuals and the reconstructed output and visually show the importance of having a combination of these to give the final output along with their frequency analysis. Although we investigated our proposal in a paired I2I setting, the idea can be extended to unpaired I2I.

5.5 Appendix

5.5.1 Implementation details

5.5.1.1 Network architecture

Our network architecture resembles a typical multi-modal I2I frameworks (*e.g.*, [213, 236]) with modifications to the decoder part of the generator to accommodate the proposed GAN Residuals. Our network consists of a U-Net [235] generator G and a style encoder E^S. The generator G consists of a content encoder E^C and a decoder with skip connections. Both the content and style encoders $\{E^C, E^S\}$ consist of 6 downsampling blocks, followed by a fully connected layer that generates a 512-dimensional latent code. Each downsampling block is a residual block borrowed from [183], with replacing average-pooling with blur-pooling. We use 64 feature maps at the first encoder layer and double this number after each downsampling block with a maximum

of 512 feature maps. The decoder network consists of 6 upsampling blocks to form a U-Net with the content encoder. The architecture of each decoder block is similar to that of StyleGAN [8], with skip connection with the corresponding downsampling block from E^C. The decoder outputs 6 channels: 3 RGB channels for the low-frequency reconstruction component $\hat{I}^B_{\text{rec}}$ and 3 channels for the GAN Residual $\hat{I}^B_{\text{res}}$. Both $\hat{I}^B_{\text{rec}}, \hat{I}^B_{\text{res}}$ are summed to form the final combined output $\hat{I}^B$. During training, we also use a discriminator network whose architecture we adapt from from [181].

5.5.1.2 Training details

We train all of our experiments and the baselines on the CelebAMask-HQ dataset [238] for approximately 200 epochs. We follow [177] and use equalized learning rate in all of our networks. We use an Adam optimizer [245] with $\beta_1 = 0, \beta_2 = 0.999$, and a learning rate of 0.001 for all networks. We linearly decay the learning rate by a factor of 10 during the last epoch of training. Our training employs three losses. First, a conditional adversarial loss $\mathcal{L}_{\text{adv}}$ where the conditional input I^A is concatenated to either the real/fake images $I^B/\hat{I}^B$ and fed to a discriminator network. Second, we use a simple L_1 loss as our reconstruction loss $\mathcal{L}_{\text{rec}}$ between our output $\hat{I}^B$ and the ground truth I^B. We use a relative weight $\lambda_{\text{rec}} = 30$ as a relative weight between the two losses. While the value of λ_{rec} was selected to bring the two loss terms to a close value range, we found the training not sensitive to the setting of this hyper-parameter. Finally, we use an L_2 regularization term on the style latent code z^s to encourage a compact latent space. For more implementation details, please refer to our code, which will be released upon acceptance.

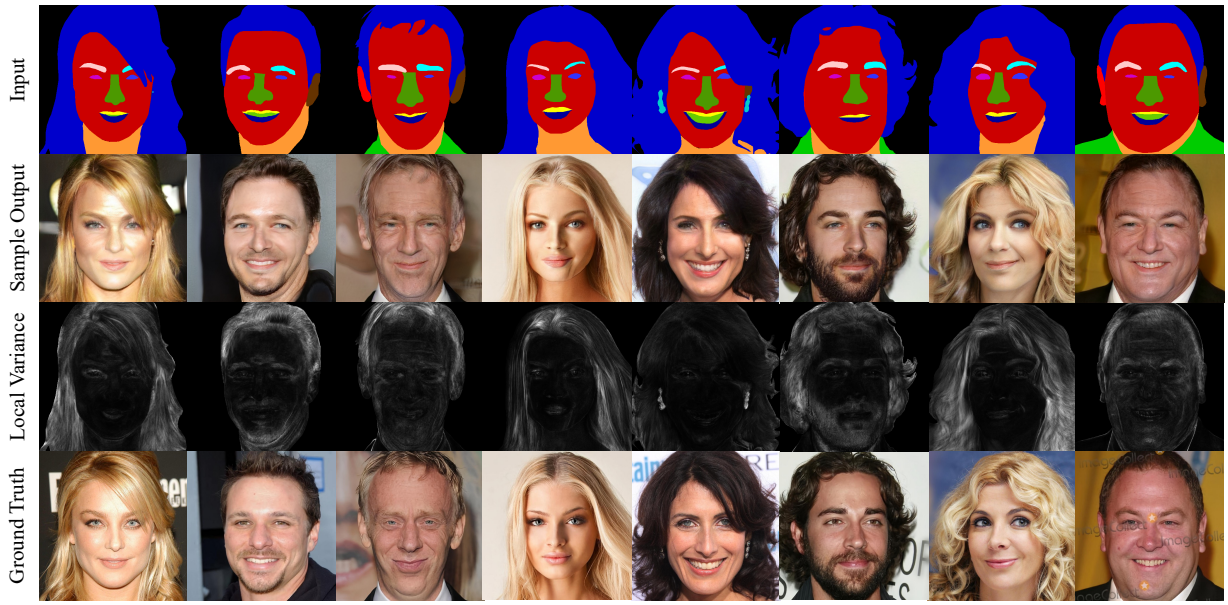


Figure 5.11: Examples of local stochastic variations. Top to bottom rows represent the input image, one sample output, the standard deviation of each pixel over 20 different outputs of the same input, and the ground truth image respectively.

5.5.2 Standard Deviation of Spatial Noise

In addition to the examples shown in the main paper, we add more examples of the standard deviation computed over diverse generations for each output image. The results are shown in Figure 5.11. As can be seen in the third row, the highest variations occur in the high-frequency regions corresponding to hair, around eyes, lips, and nose.

5.5.3 VGG vs. L1 loss

We analyze the effect of using L1 vs. VGG loss for reconstruction in Table 5.5. L1 loss is consistent with low-freq reconstruction and suits our decomposition of reconstructed and GAN residual better. As can be seen from the table, L1 is better on the PSNR, SSIM and L1 metrics. This can be attributed to the fact that L1 enforces pixel-wise reconstruction and these metrics

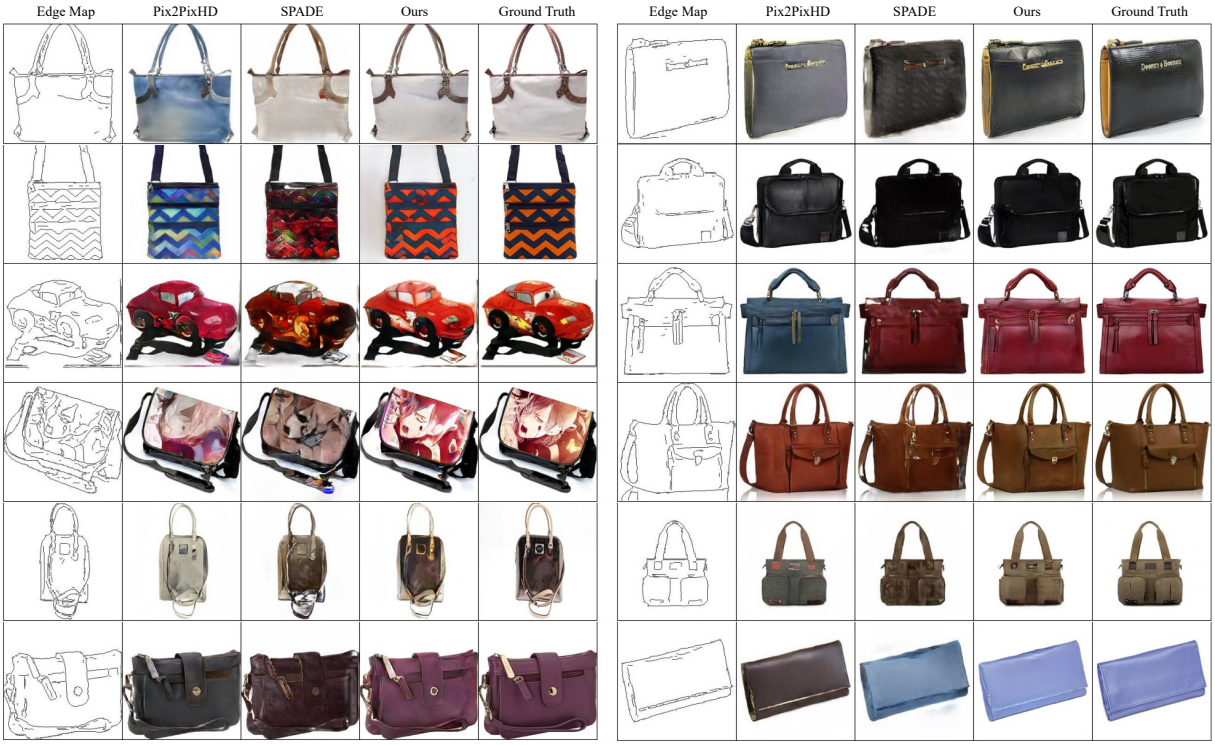


Figure 5.12: Qualitative comparison with baselines on Edges2Handbags dataset.

mostly focus on that. VGG on the other hand is a perceptual loss like LPIPS so it works better for the LPIPS metric. Also while FID is better for VGG, we visually observe it has more artifacts compared to L1.

Loss	L1 ↓	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓
L1	18.02	19.85	0.520	0.25	19.82
VGG	22.44	17.96	0.499	0.24	18.89

Table 5.5: Comparison between using L1 vs. VGG losses for supervising the reconstruction component $\hat{I}^B_{\text{rec}}$.

5.5.4 Style transfer

In Figure 5.13, we show that the network is also capable of performing style transfer. To generate these samples we generate every possible pair of translated images for 10 subjects using the images and corresponding label maps. As shown in the figure, the network is able to use the

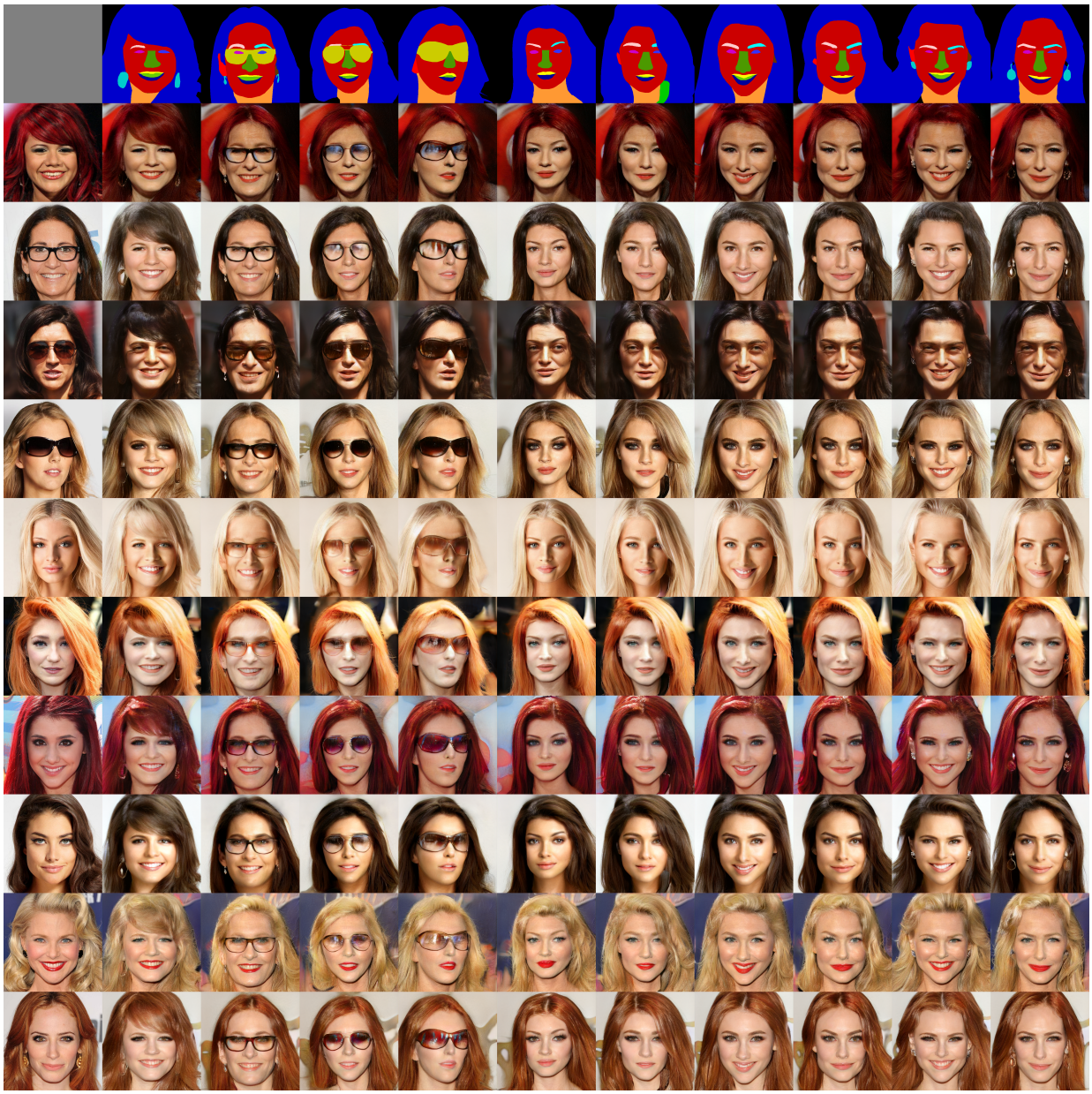


Figure 5.13: Examples of style transfer by using input label maps and style images from 10 different subjects.

style from one image and label maps from another to synthesize realistic output in most cases.

5.5.5 Qualitative results

In Figure 5.14 and Figure 5.15, we show results for qualitative comparison with [9, 229] at a 512×512 resolution on the CelebAHQ-Mask dataset. We chose these two baselines to compare

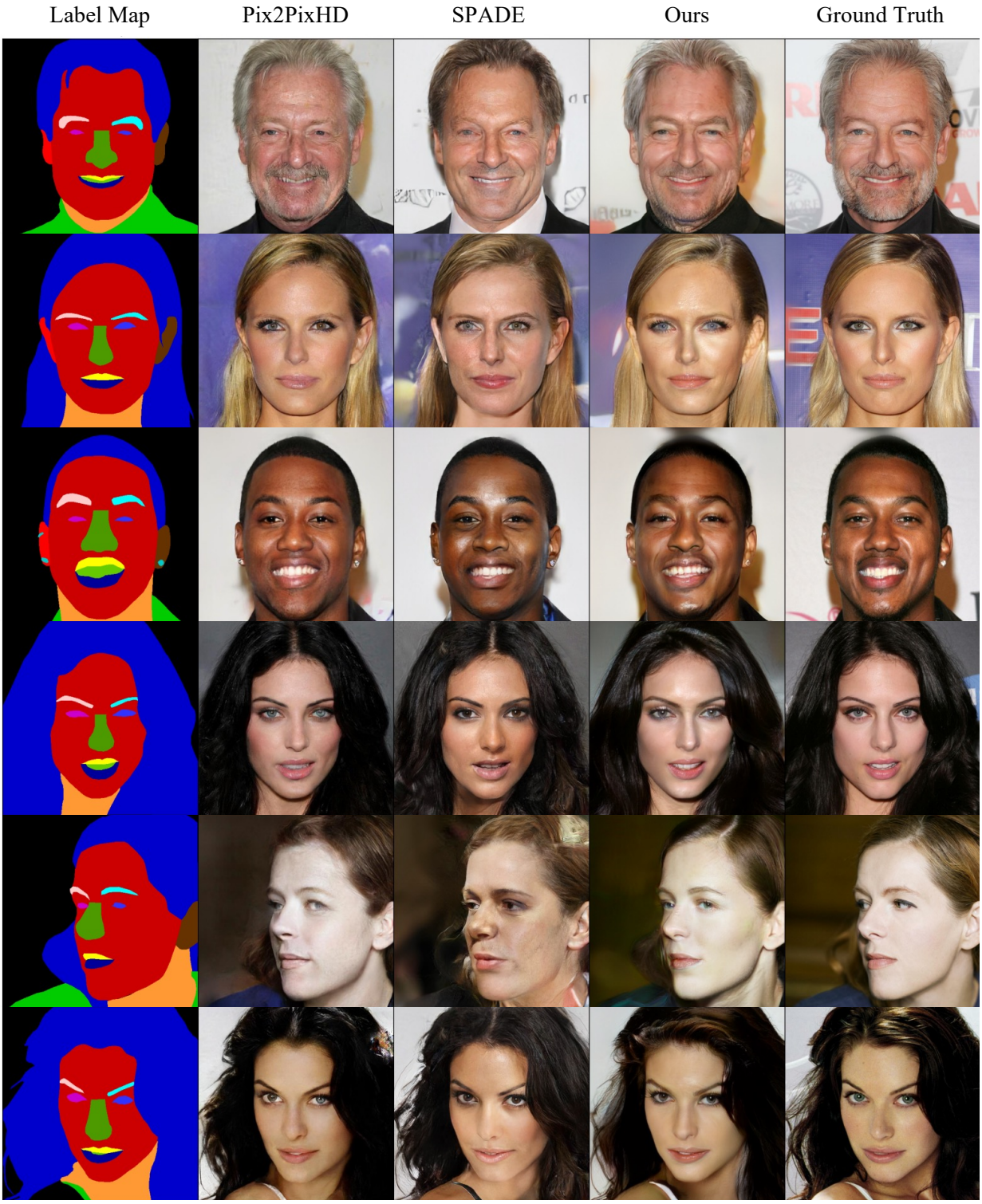


Figure 5.14: Qualitative comparisons with baselines at 512×512 resolution.

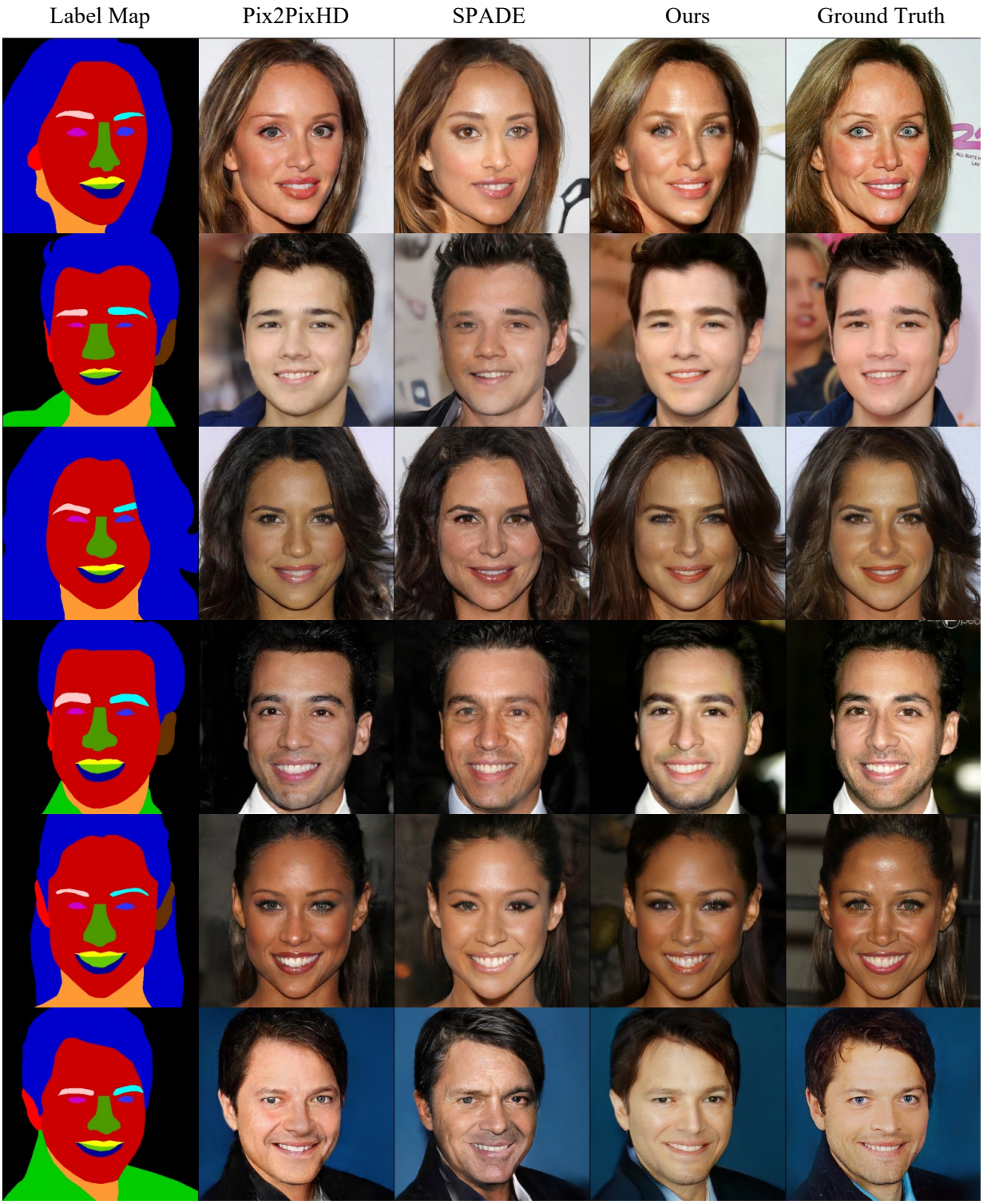


Figure 5.15: Qualitative comparisons with baselines at 512×512 resolution.

to as they were the best performing baselines at 256×256 resolution. Also in Figure 5.12 we show comparisons with the same two baselines [9, 229] on the Edges2Handbags dataset which is at 256×256. It can be clearly seen that our method generates the most realistic outputs which better matches the ground truth.

5.5.6 GAN Residuals

In Figure 5.16, we show examples of the outputs generated by our method. As reported in the paper, the reconstructed image encodes most of the structure and content of the image while the GAN residual captures the high frequency details. Combining both of them gives a realistic translation output which is close in appearance to the ground truth.

Figure 5.16: Examples of the different outputs of our method along with the input label map and ground truth image.

Chapter 6: Talking Head Synthesis

In this chapter we look at two tasks pertaining to talking head synthesis under different settings.

In the first work we study the task of learning personalized head avatars in a low-shot setting, also known as *"neural talking heads"*. Given a single-shot or few-shot images of a source subject, and a driving sequence of facial landmarks, possibly derived from a different subject, the goal is to synthesize a photo-realistic video of the source subject, under the poses and expressions of the driving sequence. This task has a wide range of applications, including those in AR/VR, video conferencing, gaming, animated movie production and video compression in tele-communication.

In the second work we look at the task of lip synchronization (lip-sync) which seeks to match the lips of human faces with different audio. It has various applications in the film industry as well as for creating virtual avatars and for video conferencing. This is a challenging problem as one needs to simultaneously introduce detailed, realistic lip movements while preserving the identity, pose, emotions, and image quality.

6.1 Learned Spatial Representations for Few-shot Talking-Head Synthesis

Traditional graphics-based approaches to this task rely on a 3D face geometry and produce very high quality synthesis. However, they tend to focus on modeling the face area without the hair, and they learn a subject-specific model and cannot generalize to new subjects. In contrast, recent 2D-based approaches [246, 247, 248, 249] learn a subject-agnostic model that can animate unseen subjects given as few as a single image. Furthermore, since these works learn an implicit model and do not require an explicit geometric representation, they can synthesize the full head, including the hair, mouth interior, and even wearable accessories like glasses and earrings. This remarkable generalization ability however comes at the cost of low quality and poor identity preservation when compared to their 3D-based subject-specific counterparts. Bridging the quality gap between 2D-based subject-agnostic and 3D-based subject-specific approaches remains an open problem.

Recent efforts in 2D-based approaches can be divided into two classes; *warping-based* and *direct synthesis*. As the name suggests, warping-based methods (*e.g.,* [247]) learn to warp the input image or a recovered canonical pose based on the motion of the driving sequence. While these methods achieve high realism, especially for static and rigid parts of the image, they tend to work well only for a limited range of motion, head rotation and dis-occlusion.

On the other hand, direct synthesis approaches (*e.g.,* [246, 248, 249]) encode the source subject into a compressed latent code, and a generator decodes the latent code to synthesize the target pose. These approaches learn a prior over the compressed latent space, and can generate realistic results for a wider range of poses and head motion. However, they exhibit a noticeable identity gap between their output and the source subject.

Figure 6.1: Our framework factorizes the image synthesis process into its spatial and style components. It predicts a discrete latent spatial layout for the target image, which is used to produce per-pixel style modulation parameters for the final synthesis.

We posit that the identity gap is caused by the entangled representation of the source subject in a single latent code. This compressed 1D latent encodes multi-view shape information, identity cues, as well as color information, lighting and background details. In order to synthesize a target view from a latent code, the generator needs to devise a complex function to decode the uni-dimensional latent into its corresponding 2D spatial information. We argue this not only consumes a large portion of the network capacity, but also limits the amount of information that can be encoded in the latent code.

To address this problem, we propose a two-step framework that decomposes the synthesis of a talking head into its spatial and style components. Our framework animates a source subject in two steps. First, it predicts a novel spatial layout of the subject under the target pose and expression. Then, it synthesizes the target frame conditioned on the predicted layout. This factorized representation yields the following key performance advantages.

Better subject-agnostic model performance. The performance of our subject-agnostic (also called *meta-learned*) model not only performs better than previous subject-agnostic state-of-the-art, but is also on-par with the subject-finetuned performance of previous works when there are only few source images available (*e.g.,* less than 10 images).

Better fine-tuned performance with less data. Fine-tuning our model for a specific subject requires significantly less data and fewer iterations than previous works, and yet achieves better performance. For example, we show that fine-tuning our model using 4-shot inputs outperforms previous state-of-the-art models fine-tuned using 32-shot inputs.

Robustness to pose variations. We show that our model is more robust against a wider range of poses and facial expressions, while still producing both realistic and identity-preserving results.

Improved identity preservation. Shape difference between the source and driving identities poses a challenge for identity preservation in reenacted results. The intermediate novel spatial representation learned by our model reduces the sensitivity towards such differences and better preserves the identity.

In summary, we make the following contributions:

- A novel approach that factorizes the *talking-head* synthesis process into its spatial and style components.
- A novel latent spatial representation that proves effective for few-shot novel view synthesis.
- We achieve state-of-the-art performance in both the single-shot and multi-shot settings, as well as in the meta-learned and subject-finetuned modes.

6.1.1 Approach

Our approach factorizes the representation of a head avatar into spatial and style components. It breaks down the novel view head synthesis of a subject into two steps. First, a layout prediction network G^l translates facial landmarks for a target view into a dense spatial layout of the subject. Then, an image generator G^s synthesizes the final image conditioned on the predicted

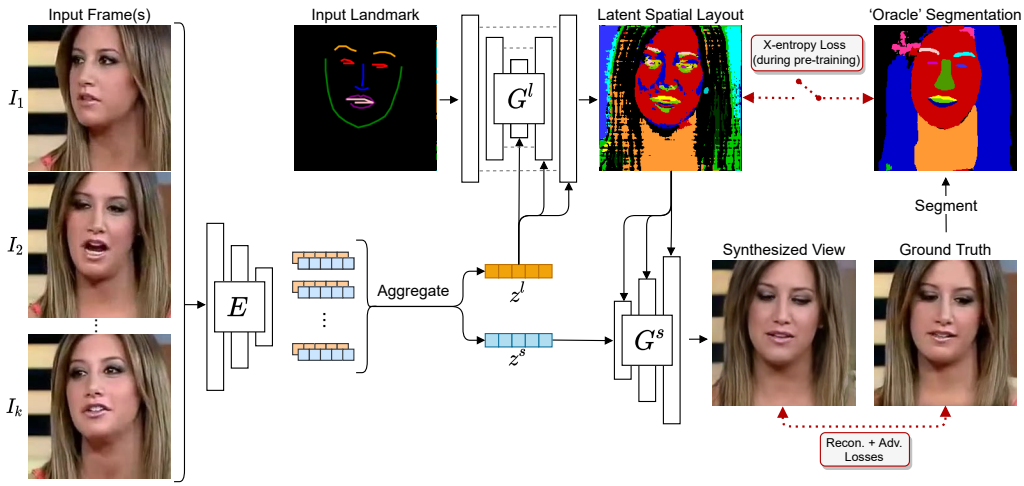


Figure 6.2: Overview of our training pipeline. The cross-entropy loss with the oracle segmentation is used during pre-training the layout predictor G^l, and then turned off during the full pipeline training.

layout. We first give an overview of our pipeline in Section 6.1.1.1. Then, we explain how to pre-train the layout prediction network G^l to predict semantic segmentations of novel views in Section 6.1.1.2, followed by the full pipeline training in Section 6.1.1.3. Section 6.1.1.4 explains how the layout prediction network G^l transitions from predicting semantic maps to learning a more powerful latent spatial representation. And finally, we discuss how to learn a personalized head avatar through an optional subject-specific fine-tuning stage in Section 6.1.1.5.

6.1.1.1 Overview

Given K-shot inputs $\{I_1 \dots I_K\}$ of a source subject, a two-headed encoder $E = \{E^l, E^S\}$ processes the inputs and generates K layout latents $\{z_i^l\}$ and K style latents $\{z_i^s\}$ for $i \in \{1 \dots K\}$. The K latents are then averaged to get an aggregated layout latent $z^l = \frac{1}{K}\sum_{i=1}^{K} z_i^l$ and style latent $z^s = \frac{1}{K}\sum_{i=1}^{K} z_i^s$. Averaging the K latents cancels out view-specific information and transient occluders, and maintains implicit 3D information like the head and hair shape for the layout latent, and color and lighting information for the style latent. We have two generators:

a layout predictor network G^l and an image generator G^s. The layout predictor takes as input the facial landmarks for a target view x_t and the layout latent z^l and generates a spatial one-hot layout $y_t^l = G^l(x_t, z^l)$, such as a semantic map, for the target view. The image generator G^s processes the style latent z^s and utilizes spatial denormalization layers (SPADE [9]), conditioned on the predicted layout y^l, to synthesize the final image $\hat{I} = G^s(y^l, z^s)$. An overview of our framework is shown in Figure 6.2.

6.1.1.2 Layout prediction pre-training

Training the above pipeline end-to-end without any supervision or constraints on the predicted layouts results in a degenerate solution, where the spatial layouts and their corresponding spatial denormalization are completely ignored. All spatial and style information are thus encoded into and decoded from the style latent z^s, which results in a poor performance. Therefore, we opted to pre-train the layout prediction network to predict a plausible semantic segmentation of a target view, given the input facial landmarks x_t and the layout latent z^l. To supervise this training, we use an off-the-shelf face segmentation network [238] as an oracle to segment the target image I_t into a semantic map S_t, and we apply a cross-entropy loss (X-ent) between the oracle segmentation S_t and our predicted segmentation $y_t^l = G^l(x_t, z^l)$. We observe that the obtained oracle segmentations are very noisy and have poor quality (*e.g.,* Figure 6.4). This is caused by the domain gap, in terms of image resolution and the distribution of head poses, between the datasets used to train the oracle segmentation network [238], and in-the-wild videos of talking heads. Thus, to regularize the segmentation prediction training, we use a mutli-task pre-training strategy where the layout prediction network predicts an extra RGB reconstruction

R_t of the target image I_t, which is used as a secondary supervisory signal. Specifically, we have

$$y_t^l, R_t = G^l(x_t, z^l), \qquad z^l = \frac{1}{K}\sum_{i=1}^{K} E^l(I_i) \tag{6.1}$$

And the objective for the pre-training is

$$\mathcal{L}_{\text{seg}} = \text{X-ent}(y_t^l, S_t) + \lambda_R \mathcal{L}_R(R_t, I_t) \tag{6.2}$$

where $\mathcal{L}_R$ is a perceptual reconstruction loss, and λ_R is a relative weighting term which is set to a low value.

6.1.1.3 Full pipeline training

Once the layout predictor network has been pre-trained to predict semantic segmentations, we plug it into our full pipeline. The predicted segmentation is fed as the spatial input to a SPADE image generator G^s that synthesizes the final image as

$$\hat{I} = G^s(G^l(x_t, z^l), z^s), \qquad z^s = \frac{1}{K}\sum_{i=1}^{K} E^s(I_i) \tag{6.3}$$

We observe that the SPADE generator quickly utilizes the input spatial segmentations to resolve spatial ambiguities, and we no longer fall into a degenerate solution where the spatial input is ignored.

Our full pipeline, comprising the layout and style encoders $\{E^l, E^s\}$, the layout predictor G^l and the image generator G^s, is optimized to minimize three losses; a reconstruction loss $\mathcal{L}_{\text{rec}}$,

an adversarial loss $\mathcal{L}_{\text{adv}}$, and a latent regularization loss $\mathcal{L}_{L2}$.

For the reconstruction loss $\mathcal{L}_{\text{rec}}$, we employ a perceptual loss [216] based on both the VGG19 [250] and VGGFace [251] networks, as well as an $L1$ loss. While the VGG19-based perceptual loss is a standard reconstruction loss, we follow Zakharov *et al*. [246] and utilize a VGGFace-based perceptual loss to promote identity preservation. We also use an $L1$ loss to better preserve color transfer between the synthesized and ground truth images.

The adversarial loss, $\mathcal{L}_{\text{adv}}$, encourages the output to be photo-realistic. To achieve that, a discriminator network D is trained to discriminate between real and fake images, while the generator network, G^s aims to fool the discriminator by bringing the output closer to the manifold of real images. We borrow the architecture of the discriminator network D from [211] and use a non-saturating logistic loss with gradient penalty [252]. Finally, we impose an $L2$ regularization on the learned latent codes to encourage compactness of the latent space. The full training objective is given by

$$\begin{aligned} \min \mathcal{L}(\hat{I}_t, I_t, z^l, z^s | E^l, E^s, G^l, G^s, D) = \mathcal{L}_{\text{rec}}(\hat{I}_t, I_t) + \\ \lambda_{\text{adv}} \mathcal{L}_{\text{adv}}(\hat{I}_t, I_t) + \lambda_{L2}\big(\|z^l\|^2 + \|z^s\|^2\big) \end{aligned} \tag{6.4}$$

where $\lambda_{\text{rec}}, \lambda_{L2}$ determine the relative weights between the loss terms.

6.1.1.4 Learning a latent spatial representation

Spatial denormalization (SPADE) generates per-pixel denormalization parameters by feeding a dense spatial input through a small convolutional subnetwork. While SPADE [9] originally

uses semantic maps as input, we explore learning a latent spatial representation that better suits the image synthesis task at hand. To do this, we turn off the cross-entropy loss so as to give the layout predictor G^l the freedom to diverge from predicting traditional semantic segmentations and learn other latent representations that better optimize the few-shot novel view synthesis objective. The layout predictor is thus supervised only by the training objective of Eqn. 6.4. Figure 6.4 shows examples of the learned latent layouts. Although they might look less interpretable than traditional semantic maps, they seem to encode more information and capture accurate details.

6.1.1.5 Subject fine-tuning

Training our full pipeline learns a powerful subject-agnostic model that produces high quality and identity-preserving synthesis. Optionally, we can learn a personalized head avatar to further refine the results for a given subject. To do this, we follow [246, 248, 249] and fine-tune the subject-agnostic model (also called *meta-learned* model) using the few-shot inputs of the source identity. Specifically, we compute the layout and style embeddings $\{z^l, z^s\}$ and fine-tune the weights of the layout and image generators $\{G^l, G^s\}$, as well as the discriminator, D, by reconstructing the set of few-shot inputs, and optimizing the same training objective of Eqn. 6.4. We observe that subject fine-tuning restores high-frequency components and improves background reconstruction when compared to the meta-learned outputs.

Figure 6.3: Qualitative comparison in the single-shot setting. We show three sets of examples representing low, medium and high variance between the source and target poses. Our method is more robust to pose variations than the baselines.

6.1.2 Experimental evaluation

Dataset. We perform our evaluation using the VoxCeleb [254] dataset, which is a large-scale in-the-wild video dataset. The train set contains over a million clips from 145,569 videos of 5,994

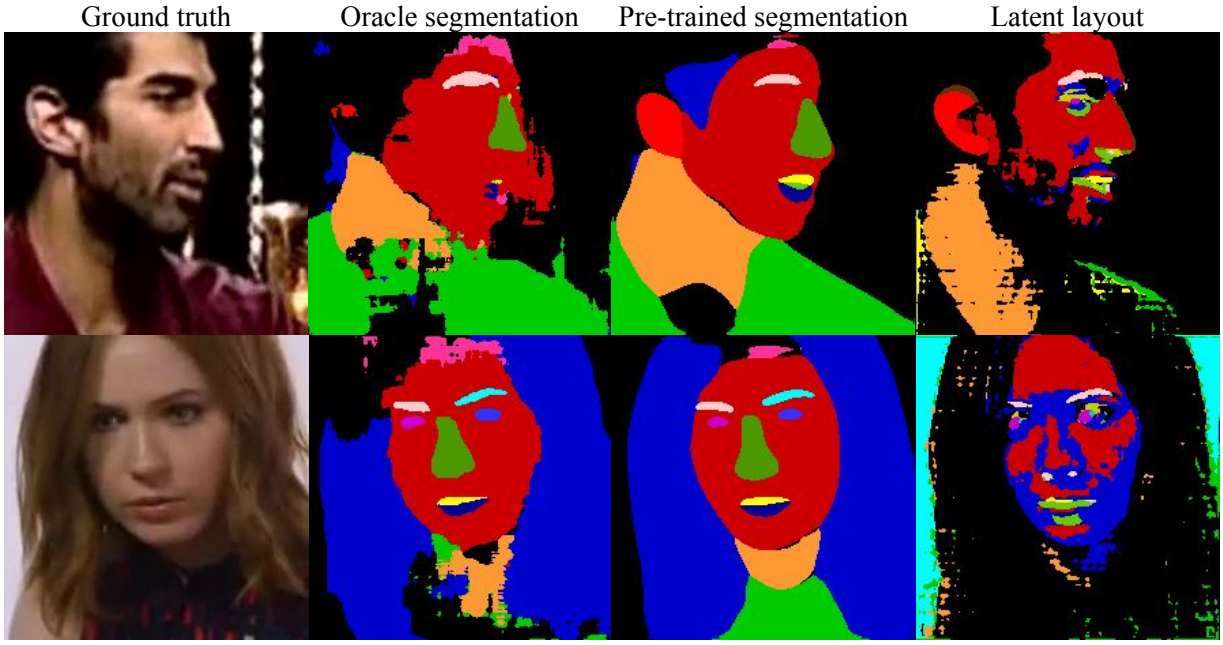


Figure 6.4: Layout pre-training predicts meaningful segmentation maps despite the noisy oracle segmentations. Our latent spatial representation encodes more information than traditional segmentations.

Table 6.1: Quantitative comparison in the single-shot setting.

Method	PSNR↑	SSIM↑	LPIPS↓	ID-SIM↑	NMKE↓	FID↓
X2Face [253]	15.50	0.466	0.346	0.691	0.333	98.58
Bi-layer [249]	–	–	–	0.721	0.236	130.58
FSTH [246]	16.92	0.597	0.263	0.836	<u>0.049</u>	53.07
LPD [248]	–	–	–	0.837	0.070	<u>48.48</u>
FOMM [247]	**18.20**	**0.635**	<u>0.236</u>	<u>0.869</u>	0.061	56.10
Ours	<u>17.37</u>	<u>0.605</u>	**0.232**	**0.886**	**0.041**	**45.69**

different identities. The test set contains new identities that are not part of the training. We use the test subset released by Zakharov *et al*. [246], which contains a total of 1,600 frames from videos of 50 subjects. For self-reenactment scenarios, the input few-shots and the driving sequence do not overlap. We obtain the facial landmarks for sampled frames using an off-the-shelf facial landmarks detector [255].

Baselines. We compare our method to the following baselines: X2Face [253], FSTH [246], FOMM [247], Latent Pose Descriptor (LPD) [248], and Bi-layer [249]. We use the released pre-trained models provided by the authors for all baselines, except for FSTH [246] where we use the authors' provided outputs, as their code and models were not released. Since some baselines only accept single-shot inputs (*e.g.,* FOMM and Bi-layer), we divide our evaluation into a single-shot setting, where we compare to all the baselines, and a multi-shot setting, where we only compare against the few-shot baselines. Since the LPD [248] and Bi-layer [249] baselines do not predict the background and re-crop the input/output frames, we subtract the background and compare with their corresponding cropped ground truths for quantitative analysis. We also exclude those two baselines from frame reconstruction evaluation since their output does not align with the rest of the methods.

Metrics. We evaluate all models along five axes.

- Reconstruction fidelity using the peak signal-to-noise ratio (*PSNR*) and structural similarity (*SSIM*) [240] metrics.
- Perceptual similarity between the output and the ground truth using the *AlexNet*-based *LPIPS* metric [241].
- Identity preservation (*ID-SIM*) using the cosine similarity between face embeddings from a face recognition network [251].
- Normalized Mean Keypoint Error (*NMKE*), which measures the pose error between the synthesized and ground truth images as computed in [248, 249].
- Perceptual quality of the output using the Frechet-Inception Distance (*FID*) metric [242].

6.1.2.1 Single-shot comparative evaluation

Table 6.1 shows a quantitative comparison with the baselines in the single-shot setting. Our method outperforms all baselines in perceptual reconstruction (LPIPS), identity preservation (ID-SIM), pose matching (NMKE) and visual quality (FID). However, FOMM scores better in the standard reconstruction metrics (PSNR and SSIM). We argue this is intrinsic to their method due to its warping-based nature, which accurately captures the background and other static regions, and thus gives low reconstruction error even in the presence of clear artifacts.

Figure 6.3 shows qualitative results from three groups representing low, medium and high variance between the input and target poses. We observe that all methods perform well when the target pose is similar to that of the input shot. LPD produces sharp results within the low-medium pose variation, but shows blurry artifacts within the face and eyes in the case of high pose variance. FSTH shows a clear identity gap. FOMM accurately matches the background and shows highly realistic results when the pose variance is low, but shows a clear identity gap and visible artifacts when the target pose is far from the source image. Our method is more robust against pose variation, yielding realistic results while preserving the source identity.

6.1.2.2 Multi-shot comparative evaluation

Here, we focus on the effect of increasing the number of K-shot inputs, and the effect of subject-specific fine-tuning using the K-shot inputs. Figure 6.6 plots the ID-SIM, NMKE and FID performance metrics as we increase the number of K-shots. We observe that the pose reconstruction performance (NMKE) is mainly dictated by the approach itself, rather than the number of K-shots or whether the models are fine-tuned or not. For example, the *meta-learning*

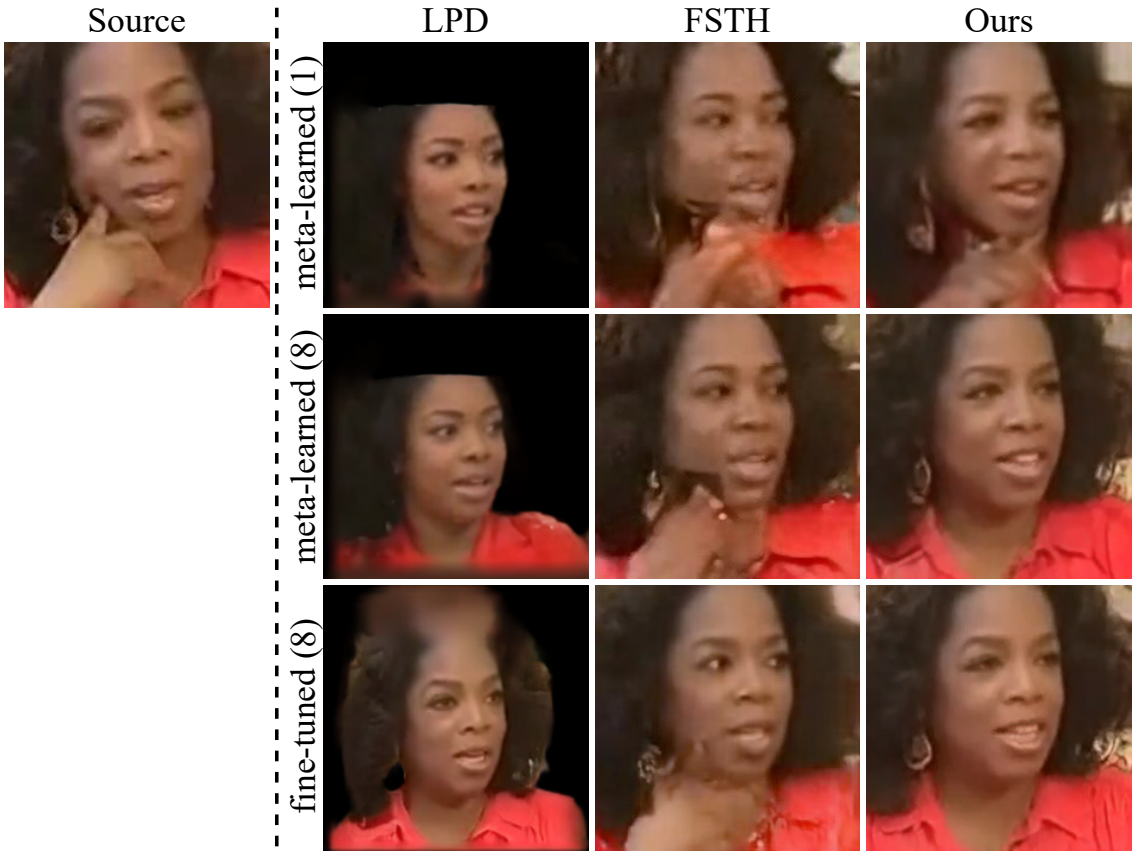


Figure 6.5: A qualitative comparison showing the effect of increasing the K-shot inputs and applying subject fine-tuning.

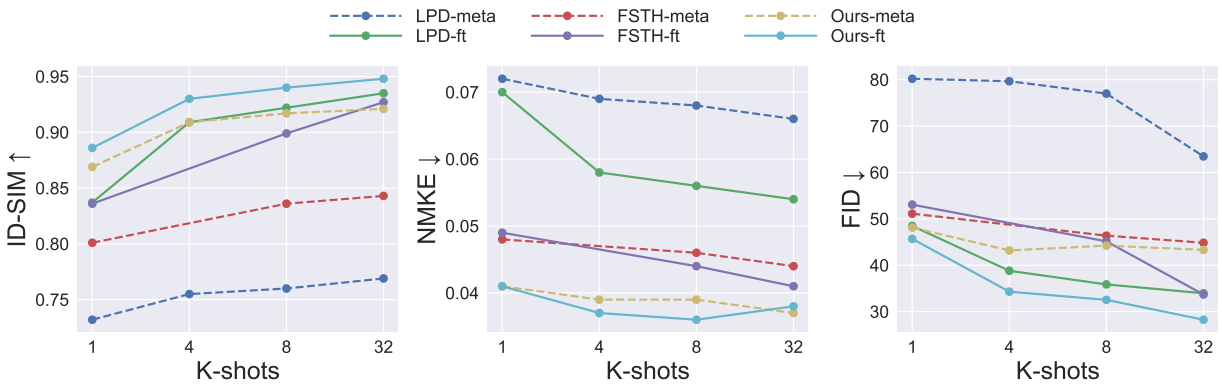


Figure 6.6: Quantitative comparison with the few-shot baselines, showing the effect of both increasing the K-shot inputs and subject-specific fine-tuning. Dotted and solid lines represent the meta-learned and fine-tuned models respectively.

performance of FSTH with $K = 1$ is better than the *fine-tuned* LPD model with $K = 32$. Similarly, the *single-shot meta-learning* performance of our method is better than the *fine-tuned*

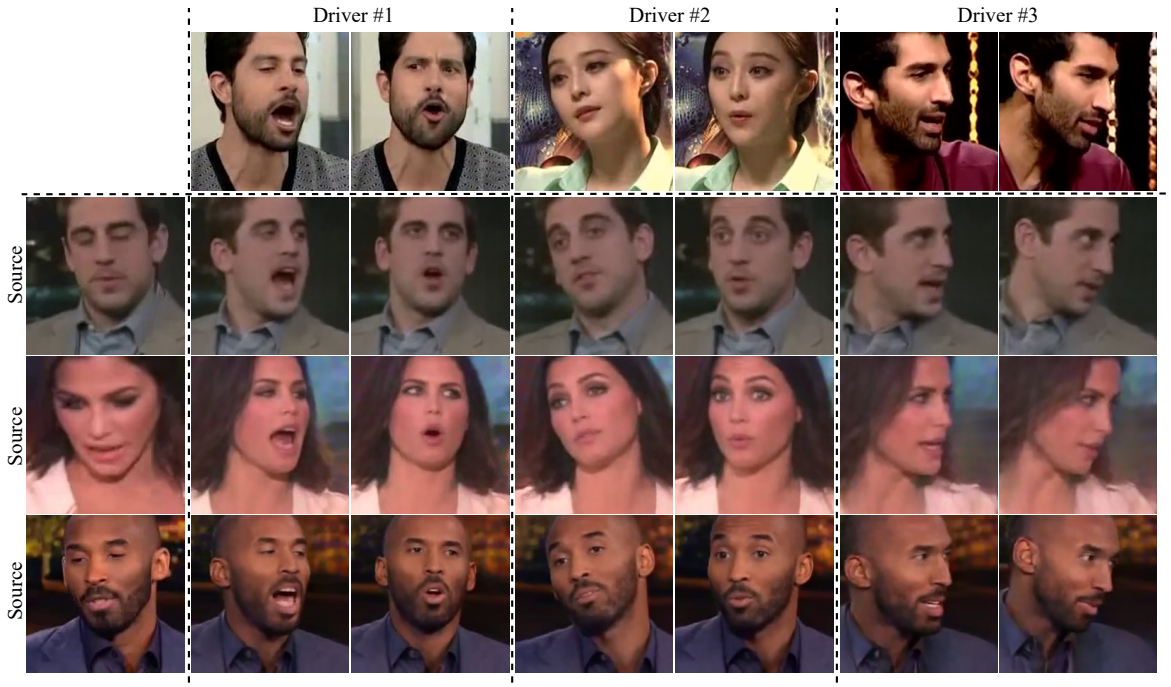


Figure 6.7: Cross-subject reenactment with different driving identities. Results are shown for our *meta-learned* model without any fine-tuning, and using 32-shot inputs.

baselines at $K = 32$.

For the ID-SIM and FID metrics, the *meta-learning* performance of our model is not only superior to that of the baselines, but it is also on-par with the *fine-tuned* baselines for $K \leq 8$. However, as K is increased to 32, the *fine-tuned* baselines eventually outperform our *meta-learned* model. Another very important advantage to our approach is that it achieves better performance with significantly less data. For example, fine-tuning our model with just $K = 4$ outperforms the fine-tuned baselines at $K = 32$. Since fine-tuning on more data requires more training iterations and thus more time, our method spends much less time fine-tuning on fewer data samples, and yet achieves similar or better results.

Figure 6.5 visualizes the effect of both increasing K and subject fine-tuning. Our method preserves the source identity without any fine-tuning, even with a single-shot input. On the other hand, the baselines only restore the source identity after the subject-specific fine-tuning. Our

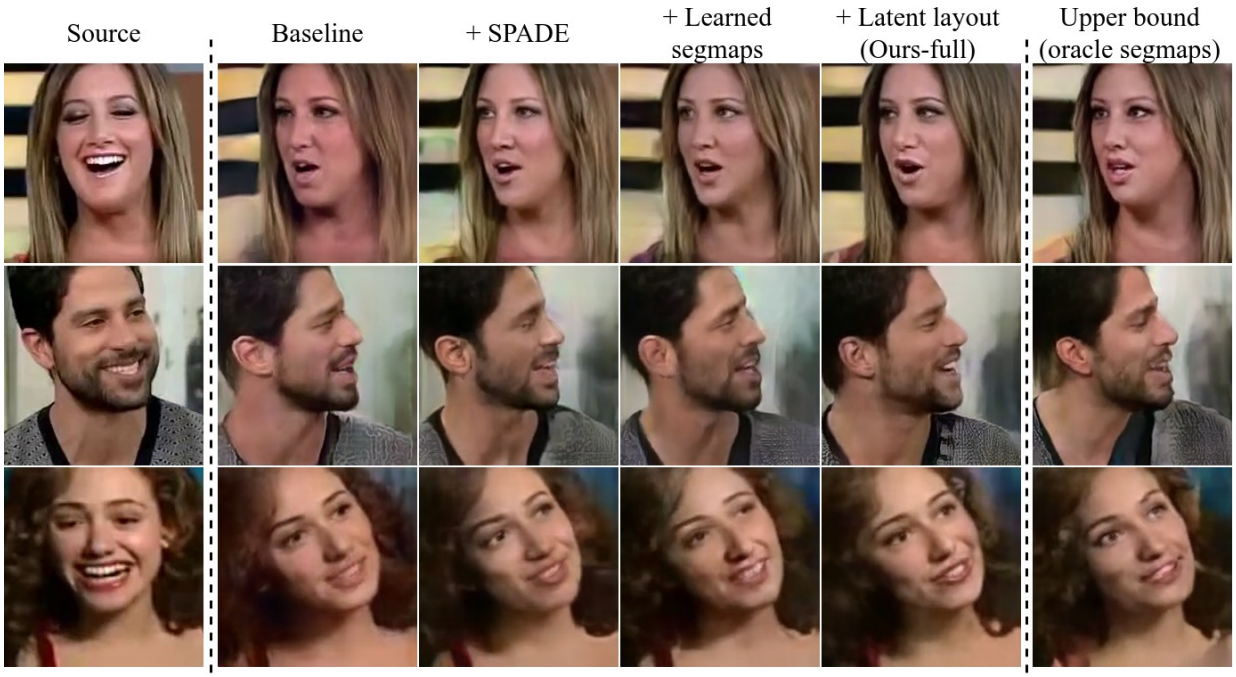


Figure 6.8: Examples from the ablation study. Results shown are for the meta-learned models with a single-shot input (source).

method also shows the most improvement, in terms of realism and better identity match, when increasing the number of K-shot inputs. For example, our method successfully filters out the subject hand occluding the face in the single-shot input.

6.1.2.3 Cross-subject reenactment

Cross-subject reenactment poses a challenge, especially for landmark-driven approaches. The shape difference between facial landmarks of the source and driver identities could lead to a noticeable identity gap in the reenacted results. The intermediate spatial representation learned by our method helps reduce this problem and leads to good identity preservation of the source subject regardless of the driver identity. Figure 6.7 shows sample reenactment results using different driver identities. To demonstrate the effectiveness of our disentangled representation, we avoid any subject fine-tuning and show the results of our meta-learned model with 32-shot inputs.

The source identity is well-preserved among challenging facial expressions and different views covering both the left and right sides of the face.

Table 6.2: Ablation study of our approach. +SPADE replaces the UNet generator with SPADE. +Learned seg. maps conditions the generator on learned segmentations. +Latent layout learns a latent spatial representation. The upper bound gets to cheat and uses the ground truth segmentations.

Method	PSNR↑	SSIM↑	LPIPS↓	ID-SIM↑	NMKE↓	FID↓
Baseline	<u>17.00</u>	0.574	0.274	<u>0.837</u>	0.044	67.19
+ SPADE	16.94	0.575	0.268	0.834	0.043	<u>56.00</u>
+ learned seg. maps	16.94	<u>0.578</u>	<u>0.265</u>	0.828	**0.042**	62.78
+ latent layout (**ours**)	**17.22**	**0.592**	**0.247**	**0.860**	<u>0.042</u>	**54.40**
Upper bound	18.21	0.629	0.219	0.867	0.039	48.06

6.1.2.4 Ablation study

We evaluate the contribution of different components of our proposed approach. All ablation experiments are trained with the same hyper-parameters and for the same number of epochs, and are evaluated in the *single-shot* setting with *no fine-tuning*. We report the results in Table 6.2. The baseline model has the same setup as FSTH [246], where a UNet generator with AdaIN layers [237] translates the input landmarks into the target image. Next, we replace the UNet architecture with a SPADE generator [9] conditioned on the facial landmarks (+SPADE). This improved the FID, but other metrics remained around the same. We hypothesize this is due to using sparse landmarks as the spatial input, while SPADE needs dense spatial inputs to generate the per-pixel denormalization parameters. To validate our hypothesis, we conducted an experiment as an *upper bound*, where we get to cheat and segment the ground truth target image using an off-the-shelf face segmentation network [238] (*i.e.*oracle), and we use these oracle segmentations

as the spatial input to SPADE. Even though the oracle segmentations are noisy (*e.g.,* Figure 6.4), this still resulted in a significant boost in all metrics, proving that the SPADE generator could benefit from dense spatial inputs. Therefore, we trained a layout prediction network to predict a plausible semantic segmentation for the target pose (+Learned seg. maps). This surprisingly produced mixed results and even caused a drop in the ID-SIM and FID scores. We posit this is because the noisy oracle segmentations do not provide a consistent supervisory signal, which causes the learned segmentations to miss important shape cues (*e.g.,* the correct face shape), as well as overfit common errors in the oracle segmentations as the training progresses. Finally, removing the supervision on the predicted layouts and learning a latent spatial representations (+Latent layouts) resulted in a reasonable performance improvement over all metrics. We also show a qualitative comparison for the ablation study in Figure 6.8. We observe that the qualitative results of the upper bound experiment (using the oracle/ground-truth segmentation) exhibits artifacts caused by errors in the oracle segmentation. The results of our method with the learned latent layouts looks qualitatively better, with no clear artifacts, despite having worse quantitative metrics than the upper bound experiment.

6.2 Diff2Lip: Audio Conditioned Diffusion Models for Lip-Sync

Oscar-winning director Bong Joon Ho famously pointed out that subtitles act as a barrier between a foreign audience not adept in the language and their ability to fully enjoy amazing movies [256], as the viewer needs to focus on both watching and reading. A rarely explored alternative, multiple-language version movie (MLV), where the same film is shot in multiple languages in parallel, is naturally much more expensive [257]. While dubbing is a popular com-

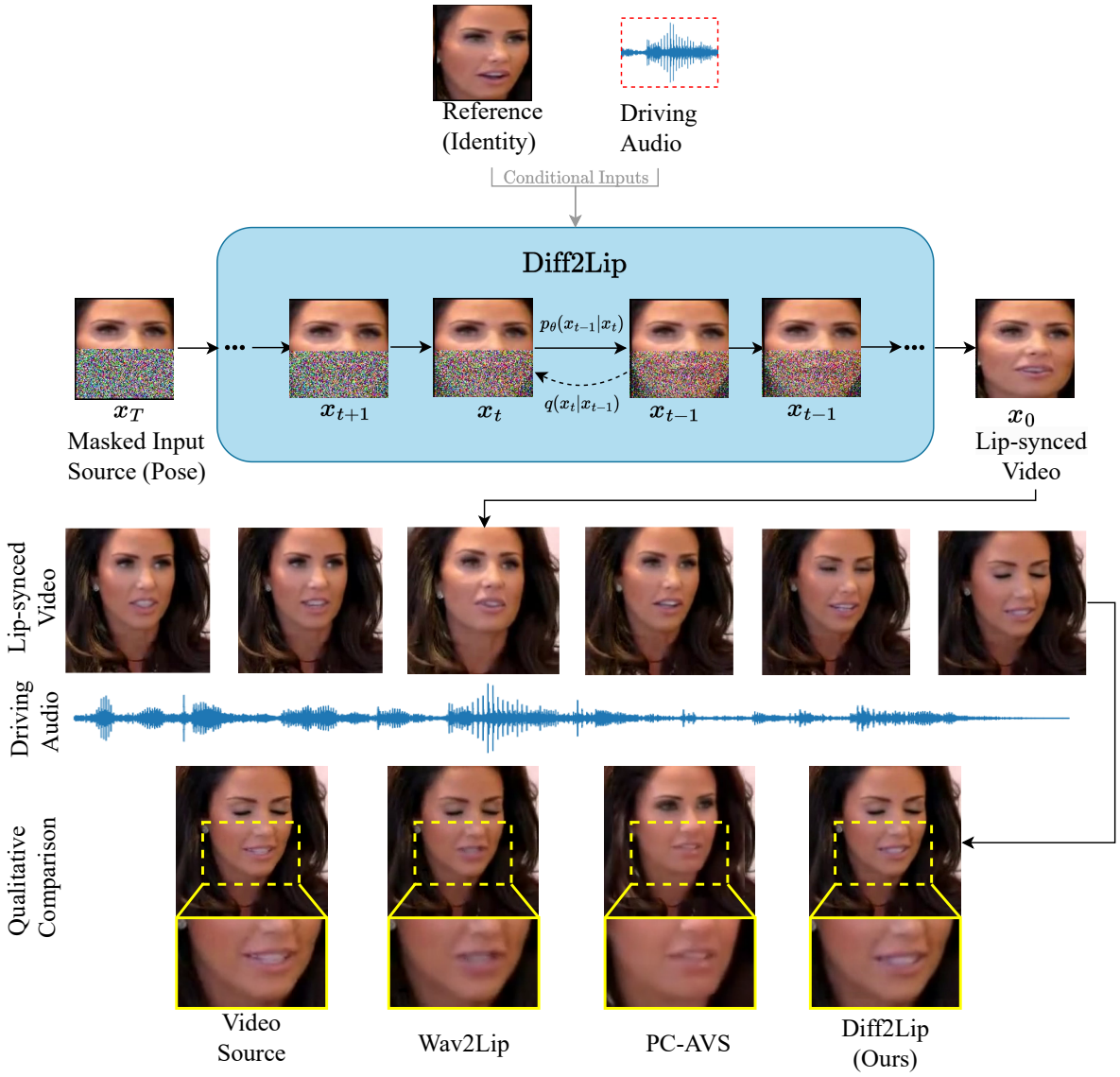


Figure 6.9: **Top:** Our Diff2Lip approach uses an audio-conditioned diffusion model to generate lip-synchronized videos. (Here q denotes the forward diffusion process and p_θ is the learned reverse diffusion process.) **Bottom:** On zooming in to the mouth region it can be seen that our method generates high-quality video frames without suffering from identity loss.

promise solution, it can feel unnatural due to the lack of synchronization between speech and actors' video.

As a cheaper alternative, lip-synchronization (lip-sync) aims to generate the mouth region of the human face such that the lips correspond to a different speech audio. Its applications beyond movies include education, virtual avatars, video conferencing, assistive technology, and

culture preservation. Ideally, lip-sync should support any identity and audio from unseen sources (in-the-wild).

This brings up the challenges of preserving the actors' identity, pose, emotions, and visual quality while maintaining a realistic lip-sync.

One of the earliest lip-sync methods, Video Rewrite [258], had a purpose-built solution by mapping phonemes to mouth shapes and then blending them onto the target video. Modern techniques have more general solutions but suffer certain limitations. For instance, PC-AVS [14] and GC-AVT [259], disentangle pose and expression respectively but fail to preserve identity (Fig. 6.9 bottom), have worse visual quality, and have border inconsistencies (while putting the generated heads back to the scene). On the other hand, works that target a specific identity, such as SynthesizingObama [260], require video/identity-specific training. Other methods which rely on extracting intermediate representations, e.g., landmarks in MakeItTalk [261], have to deal with estimation errors in these representations. Finally, approaches that can generalize on in-the-wild lip-sync settings pose it as an inpainting task, where the mouth region is masked and then generated according to the audio. Examples include Wav2Lip [13], which achieves good lip-sync but at the cost of poor visual quality (see Fig. 6.9 bottom), and AV-CAT [262], which has a multistage pipeline but does not capture finer details. In this paper, we introduce Diff2Lip, an inpainting style approach that solves the lip-sync task using diffusion models, which addresses most of these shortcomings and achieves visually superior lip-sync results.

We propose an audio-conditioned diffusion model to solve the task of lip-sync (Fig. 6.9 top). Diffusion models [263] are likelihood-based models that can generate astonishing results in high variation datasets (e.g., ImageNet [264]), that GANs [265] cannot match. To generalize in-the-wild, we pose the problem as a conditional diffusion model based inpainting task [266].

Diff2Lip takes three inputs: a masked input frame, a reference frame, and an audio frame, and outputs the lip-synced mouth region. Diff2Lip leverages (1) the masked input frame to get the pose context; (2) the reference frame to get the identity and mouth region textures; (3) the audio frame to drive the lip shape.

Using an audio+image conditioned diffusion model, Diff2Lip maintains a fine balance between all these contextual input information, avoiding lip-sync problems (e.g. identity loss, reference copying, inaccurate lip shape). Diff2Lip optimizes three losses: a reconstruction loss to guide synthesis; a sync-expert loss [13] to enforce synchronization; and a sequential adversarial loss to enforce inter-frame continuity.

Diff2Lip generates high image quality without identity loss or generalizability issues as shown in Fig. 6.9 bottom.

We evaluate our work on commonly used benchmarks of Voxceleb2 [6] and LRW [7] datasets for the tasks of reconstruction and cross generation (see section 6.2.2). We compare against popular methods used for lip-sync like Wav2Lip [13] and PC-AVS [14]. Extensive evaluations show that Diff2Lip outperform existing methods in terms of image fidelity while having comparable synchronization.

The following are the contributions of this work:

- We propose a novel diffusion model based approach for audio-conditioned image generation.
- Using frame-wise and sequential losses we are able to successfully generate high quality lip-sync.
- We show that the use of a sequential adversarial loss makes frame-wise video generation more stable for diffusion models across frames.

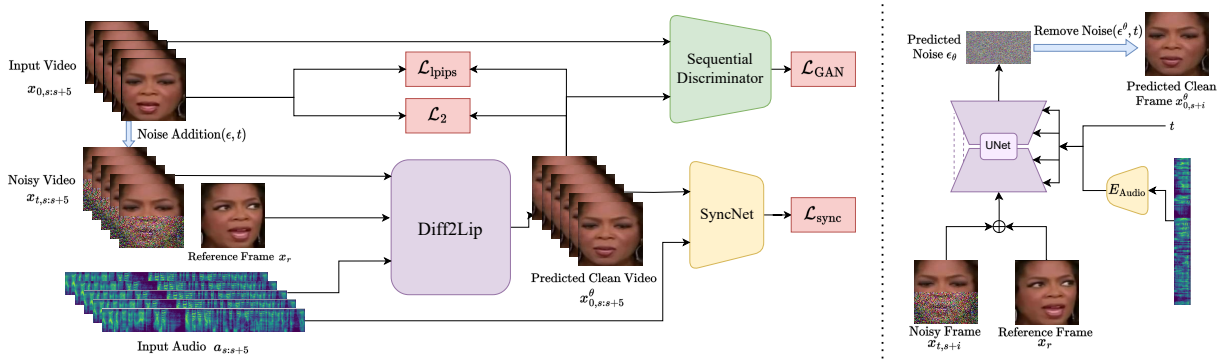


Figure 6.10: **Overview:** Diff2Lip solves lip-sync using an audio-conditioned diffusion model, which learns to inpaint the lower half of the face. During training (left), given an input video sequence $x_{0,s:s+5}$, we first add noise to the lower half using the forward process (Eq. 6.6) to get the noisy video sequence $x_{t,s:s+5}$, where diffusion step t is sampled uniformly. Then a noisy video frame $x_{t,s+i}$ for $i \in [0, 5)$, a different random reference frame x_r, and the audio frame a_{s+i} is input to our model. The audio encoder E_{Audio} encodes the audio frame a_{s+i}. Our model (right), predicts the added noise ϵ_θ given these inputs, which is used to get the predicted clean frame $x^{\theta}_{0,s+i}$ (using Eq. 6.6). Then frame-wise reconstruction losses like $\mathcal{L}_2$ and $\mathcal{L}_{\text{lpips}}$ are applied to the predicted clean sequence $x^{\theta}_{0,s:s+5}$ for enforcing good image quality while sequential losses like sequential adversarial loss $\mathcal{L}_{\text{GAN}}$ and SyncNet expert loss $\mathcal{L}_{\text{sync}}$ ensure lip-sync.

- Extensive evaluations validate that our generations outperform existing methods in FID metric and MOS of the users showing the effectiveness of Diff2Lip.

6.2.1 Approach

In this section, we discuss our proposed approach - Diff2Lip. We propose a novel audio and image conditioned diffusion model which is able to synthesize high quality lip-synced mouths corresponding to the audio input.

6.2.1.1 Diffusion Models

Diffusion models [263] are likelihood-based models which try to sample points from a given distribution by gradually denoizing random gaussian noise in T steps. In the forward diffusion process, increasing amounts of noise is added to a sample point x_0 iteratively as $x_0 \rightarrow$

$x_1 \to \cdots \to x_{t-1} \to x_t \to \cdots \to x_{T-1} \to x_T$, to get a completely noisy image x_T. Formally, the forward diffusion process is a Markovian noising process defined by a list of noise scales $\{\bar{\alpha}_t\}_{t=1}^{T}$ as:

$$q(x_t|x_0) := \mathcal{N}(x_t|\sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbf{I}) \tag{6.5}$$

which can be rewritten as:

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, \epsilon \in \mathcal{N}(0, \mathbf{I}) \tag{6.6}$$

where ϵ is the noise, $\mathcal{N}$ denotes normal distribution, x_0 is the original image, and x_t is noised image after t steps of the diffusion process. The reverse diffusion process aims to learn the posterior distribution $q(x_{t-1}|x_0, x_t)$, using which one can estimate x_{t-1} given x_t. This is typically done using a neural network, which can be parameterized in multiple ways. Similar to [263, 267, 268], we choose to parameterize the neural network to predict the noise, ie. $\epsilon_\theta(x_t, t)$, where θ represents the parameters of the neural network. It takes a noisy sample x_t and timestep t to predict the added noise ϵ in Eq. 6.6. The model is learned using the simplified objective used in [263] which reweights the variational lower bound on the maximum likelihood objective:

$$\mathcal{L}_{\text{simple}} = \mathbb{E}_{x_0,t,\epsilon}[\|\epsilon_\theta(x_t, t) - \epsilon\|_2^2] \tag{6.7}$$

The posterior distribution $q(x_{t-1}|x_0, x_t)$ is also tractable using the Bayesian rule and turns out to be another normal distribution. When using DDIM [269] for sampling, we can deterministically sample the posterior by disregarding the variance. Since we can write x_0 in terms of x_t

and ϵ using Eq. 6.6, therefore we can recover x_{t-1} deterministically given x_t and ϵ using:

$$x_{t-1} = \sqrt{\frac{\bar{\alpha}_{t-1}}{\bar{\alpha}_t}} x_t + \left(\sqrt{1 - \bar{\alpha}_{t-1}} - \sqrt{\frac{\bar{\alpha}_{t-1}(1 - \bar{\alpha}_t)}{\bar{\alpha}_t}} \right) \cdot \epsilon \tag{6.8}$$

This equation represents the mean of the learned posterior $p_\theta(x_{t-1}|x_{x_t})$ distribution in the DDIM [269] formulation.

For sampling during inference time, x_T is sampled from the standard normal distribution. The neural network can then recover the noise ϵ_θ that needs to be removed. This in turn can be fed into Eq. 6.8 to get back x_{T-1}. Iterating over this one can get the clean image as $x_T \rightarrow x_{T-1} \rightarrow \cdots \rightarrow x_t \rightarrow x_{t-1} \rightarrow \cdots \rightarrow x_1 \rightarrow x_0$ as seen in Fig. 6.11 top.

Notation. In this paper, we work with diffusion processes and videos. We use t for the diffusion process step number while s for the video frame number. For the diffusion process, we keep the notation here the same as [267].

6.2.1.2 Proposed Approach

We pose the problem of lip-sync as a lower mouth inpainting task, where given an input face with the lower half masked, an audio frame input, and a reference frame input, the model needs to generate the masked region of the face. Formally, given a video $V = \{v_1, \ldots v_S\}$ with S frames, were v_s is the s^{th} frame, and audio $A = \{a_1, \ldots, a_S\}$, where a_s is the s^{th} audio frame, our model processes one video frame $x_{0,s} = v_s$ at a time. Let $x_{s,T} = v_s \cdot (1 - M) + \eta \cdot M$ be a noise-masked video frame, where $\eta \in \mathcal{N}(0, \mathbf{I})$ and M is a binary mask for the lower half of the face. (Here the subscript T denotes a completely noised frame that we want to denoise). We want our trained model to be able to recover v_s using the reverse diffusion process, given inputs

Figure 6.11: Intermediate x_t (**top**) and x_0^θ (**bottom**) as t goes from T to 0 (left to right), sampled at uniform intervals.

masked video frame $x_{s,T}$, the audio frame a_s, and a random reference frame $x_r = v_{\text{random}(1,S)\neq s}$. This setup is quite similar to Wav2Lip[13]. The random reference frame x_r is chosen from the same video and provides cues about the source's identity and pose. We make sure that it is not the same as the input frame; otherwise there could be information leakage while training. The audio input a_s provides information about the lip structure.

As shown Fig. 6.10 we formulate the problem as an inpainting task similar to [266], i.e., we learn a conditional model $\epsilon_\theta(x_{s,t}, a_s, x_r, t)$. At training time, we take a clean sample frame $x_{s,0}(= v_s)$ and a uniformly sampled t, and add noise to $x_{s,0}$ using Eq. 6.6 to get $x_{s,t}$. The model is trained to predict the noise $\epsilon \in \mathcal{N}(0, \mathbf{I})$ added to it using Eq. 6.7.

We feed the reference frame by concatenating it with the input frame while the audio is fed using group normalization (similar to time and class conditioning in [267]). Our network has a UNet [270] backbone which consists of residual blocks and attention blocks similar to [268]. We want the UNet to extract contextual information from the unmasked portion of the input frame, and the reference frame. To enforce this we provide these directly as input to the network. For the audio which is used as a conditioning, we first encode it using a trainable encoder E_{Audio}, which generates embeddings that are injected as conditioning. E_{Audio} is also built using the same blocks as the UNet.

6.2.1.3 Additional Losses

When just training using $\mathcal{L}_{\text{simple}}$ (applied to the masked region), we observe that the mouth region generation had good image quality but no lip-sync. Hence we add additional losses to make our model work.

Our model predicts in noise space and hence many image-space losses cannot be directly applied to it. There are three ways to approach this issue - first, our model could be parameterized to directly predict the denoized image x_0 instead of predicting ϵ. Second, we can use the sampling process described in Section 6.2.1.1 to recover back the clean image x_0. Third, substituting x_t and ϵ_θ into Eq. 6.6 one could directly recover $x_0^\theta(x_t, t)$, an estimate of x_0, without having to do iterative sampling. We observed that directly predicting denoized image leads to worse image quality while using iterative sampling is overly time-consuming and hence we stick with predicting ϵ.

This approach leads to a noisy $x_0^\theta(x_t, t)$ when the step t is large as seen in Fig. 6.11 bottom but there have been previous works [271] which have applied image losses to $x_0^\theta(x_t, t)$. We enforce an $\mathcal{L}_2$ loss on this $x_0^\theta(x_t, t)$ to make it clean:

$$\mathcal{L}_2 = \mathbb{E}_{x_{0,s},t,\epsilon}[\|x_{0,s}^\theta - x_{0,s}\|_2^2] \tag{6.9}$$

Next, to impose audio synchronization, we use SyncNet discriminator as used by Wav2Lip [13]. We first separately train the SyncNet in a contrastive manner which is kept fixed during training our generation model. Similar to Wav2Lip [13] we work with a sequence of 5 frames as input to the SyncNet. By using 5 predicted video frames $x_{0,s:s+5}^\theta$ and the corresponding audio sequence

$a_{s:s+5}$, SyncNet loss can be written as:

$$\mathcal{L}_{\text{sync}} = \mathbb{E}_{x_{0,s},t,\epsilon}[\text{SyncNet}(x^{\theta}_{0,s:s+5}, a_{s:s+5})] \tag{6.10}$$

As shown in our ablation, directly adding SyncNet loss, deteriorates the image quality. To mitigate this we add perceptual similarity loss [272] on the generated frames:

$$\mathcal{L}_{\text{lpips}} = \mathbb{E}_{x_{0,s},t,\epsilon}\mathbb{E}_{l}[\|\phi_l(x^{\theta}_{0,s}) - \phi_l(x_{0,s})\|^2_2] \tag{6.11}$$

where $\phi_l(\cdot)$ represents the features coming from the l^{th} layer of a pretrained-VGG network. Finally, to enforce temporal consistency we also add a sequence adversarial loss. This makes the movement of the lips realistic across frames.

$$\begin{aligned}\mathcal{L}_{\text{GAN}} = \mathbb{E}_{x_{0,s},t,\epsilon}[\log D_{\psi}(x^{\theta}_{0,s:s+5})]+ \\ \mathbb{E}_{x_{0,s}}[\log(1 - D_{\psi}(x_{0,s:s+5}))]\end{aligned} \tag{6.12}$$

where we use a PatchGAN [273] discriminator D_{ψ}. This task requires more context than just two frames [274] but no optical flow [275]. The overall optimization objective can be written as:

$$\begin{aligned}\mathcal{L} = \mathcal{L}_{\text{simple}} + \lambda_{\text{l2}}\mathcal{L}_2 + \lambda_{\text{sync}}\mathcal{L}_{\text{sync}}+ \\ \lambda_{\text{lpips}}\mathcal{L}_{\text{lpips}} + \lambda_{\text{gan}}\mathcal{L}_{\text{GAN}}\end{aligned} \tag{6.13}$$

For sequence-based losses, it is essential that the diffusion process step input t is the same for a sequence of frames $x_{0,s:s+5}$. This ensures uniformity within a predicted sequence during loss computation.

Table 6.3: Ablation over our losses (Reconstruction)

Losses	FID ↓	SSIM ↑	PSNR ↑	LMD ↓	Sync_c ↑
Reconstruction	8.589	0.523	18.234	3.472	0.633
+ SyncNet	8.998	0.526	**18.57**	3.123	6.336
+ Perceptual	**7.751**	0.526	18.548	3.121	6.53
+ Seq. GAN	8.213	**0.527**	18.52	**3.101**	**7.89**

6.2.2 Experiments

Datasets. We evaluate our method on the Voxceleb2 [6] and LRW [7] datasets, which contain in-the-wild videos of talking human faces and are commonly used for lip-sync research.

Voxceleb2 [6] - consists of over 1M face-cropped Youtube videos coming from 6000+ identities. This dataset consists of high variation in lighting, image quality, pose, and motion blur. The average video length is 8 seconds.

LRW [7] - is a lip-reading dataset that contains 1000 videos each of 500 different words for a length of 1 second coming from BBC news. It has less variation compared to Voxceleb2 and focuses on clean front-facing videos. Like previous works, our model is trained only on the Voxceleb2 train split while we test on both datasets. We don't use the whole dataset for training but rather only use the first utterance of every video, which totals 145K videos.

Implementation Details. We preprocess the videos to have a framerate of 25 fps and an audio sample rate of 16kHz. For all our models the video resolution is 224×224 out of which we crop the face and resize it to 128×128. This is then masked in the lower half using gaussian noise and fed to our model which only morphs the lower half of this image according to the audio input. Then we resize it back to the original crop size and place it back on the video. For audio inputs, we first sample the audio at 16kHz and then create mel-spectrograms with window-size

Table 6.4: Quantitative comparison with baselines on Voxceleb2 [6] and LRW [7] on the task of reconstruction and Cross generation.

Dataset	Method	Reconstruction						Cross			
		FID ↓	SSIM ↑	PSNR ↑	LMD ↓	Sync_c ↑	Sync_d ↓	FID ↓	LMD ↓	Sync_c ↑	Sync_d ↓
VoxCeleb2	Wav2Lip [13]	3.26	0.53	18.18	3.16	**9.08**	5.93	5.11	4.84	**8.12**	6.74
	PC-AVS [14]	4.25	0.53	**18.26**	3.16	6.71	7.80	10.62	5.00	6.96	7.53
	Diff2Lip (Ours)	**2.46**	0.53	18.09	**3.04**	8.78	**5.93**	**4.53**	**4.82**	7.62	**6.73**
LRW	Wav2Lip [13]	4.23	**0.68**	**20.76**	**2.15**	**8.13**	**6.09**	5.19	**3.88**	**7.52**	**6.56**
	PC-AVS [14]	6.80	0.61	20.10	2.29	6.68	7.29	8.48	4.09	6.66	7.27
	Diff2Lip (Ours)	**2.62**	0.67	20.62	2.17	7.41	6.21	**2.54**	3.93	6.44	6.97

800 and hop-size 200. These audio frames turn out to have size 16×80. We build our code on top of the guided-diffusion repository [268]. We train our model on 8 NVIDIA RTXA6000 GPUs which takes around 4 days. Our model is trained using $T = 1000$ diffusion steps, but for faster inference, we use only 25 steps of DDIM [269] sampling which takes 4.67 seconds on an average for all the frames of one VoxCeleb2 [6] video (avg. 8 seconds at 25 fps) on 8 NVIDIA RTXA6000 GPUs.

Comparison Methods. We compare our method against the most popular methods for lip-sync. Our choice of models is based also on models/codebases which are publicly available. Wav2Lip [13] is an inpainting style method that uses SyncNet expert loss to get good lip-sync. PC-AVS [14] is a head reconstruction method that focuses on controlling pose apart from identity and lip shape. For both these methods we use their publicly available pre-trained models for the evaluation of all the datasets.

6.2.2.1 Quantitative Evaluation

For quantitative evaluation, we evaluate our model in terms of both the visual quality as well as audio synchronization. For visual quality, we use FID [276], SSIM [277], and PSNR, which

Table 6.5: Ablation over our losses (Cross generation)

Losses	FID ↓	LMD ↓	Sync_c ↑
Reconstruction	6.694	5.313	0.992
+ SyncNet	8.784	**4.816**	5.946
+ Perceptual	5.016	5.009	5.955
+ Seq. GAN	**4.592**	4.985	**6.83**

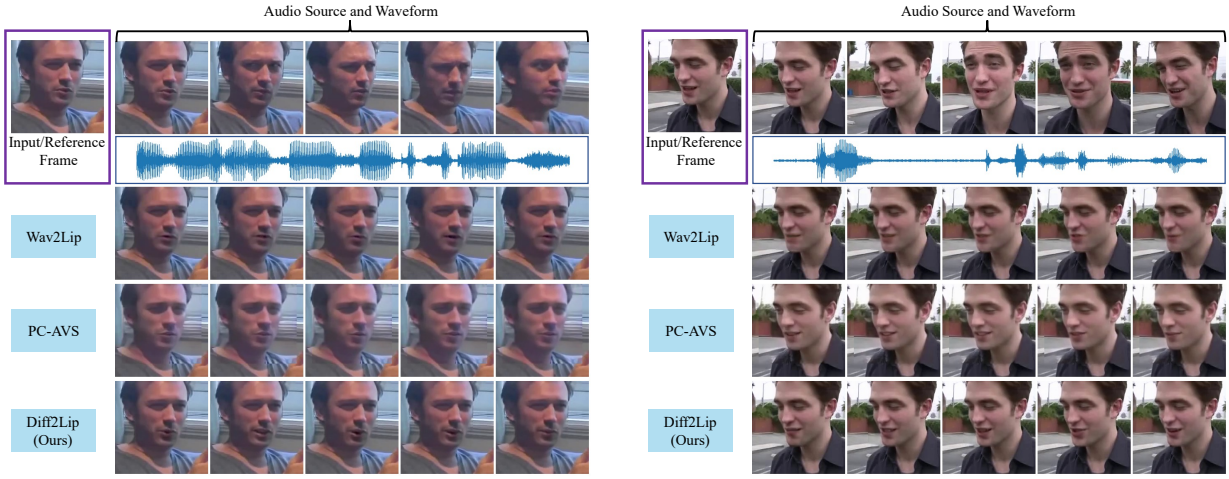


Figure 6.12: **Qualitative results of Reconstruction on VoxCeleb2 [6]**. Here we provide only the first frame as the input source (for pose) as well as the reference frame (for identity), and this frame is driven using the audio (second row) coming from the same video (top row). Wav2Lip [13] blurs the lip region in both cases to achieve the correct lip shape while PC-AVS has identity loss (see right) and border discontinuity. Our generations look highly realistic and have the lip shapes as in the audio source. (Please zoom in for better visibility.)

are popular metrics used in papers like Wav2Lip [13], PC-AVS [14], AV-CAT [262]. FID is a popular metric used for comparing the "realness" of generated images by comparing against the real image distribution. SSIM and PSNR are pixel-wise image similarity metrics that compare a pair of images and are not suited for capturing variability in video generation [278] but are included in this work for completeness. While to measure synchronization we use LMD [279], Sync_c, and Sync_d [280]. LMD measures the distance between mouth landmarks among frames. Sync_c is the confidence score of SyncNet while Sync_d is the average distance between SyncNet video and audio representations, which tell the synchronization quality. Note that for evaluation,

Figure 6.13: **Qualitative results of Cross generation on VoxCeleb2 [6]**. Here we provide a video source (first row) and drive that identity-pose combination using audio coming from a different video (second and third rows). Wav2Lip [13] blurs its generations, for example, beard region details are missing on the right. PC-AVS's [14] generations have flaws like identity loss, in both cases. They introduce artifacts near the eyes on the left while there is identity loss on the right. Our method generates realistic mouths with expressive lips while being in sync with the audio source. In the bottom 2 rows, we can see that the lip region of our generations match those of the audio source. (Please zoom in for better visibility.)

we use the pre-trained SyncNet from the SyncNet's [280] repository but for training, we train our own SyncNet, similar to Wav2Lip[13] and AV-CAT[262]. We use pre-trained models of Wav2Lip and PC-AVS methods to conduct our evaluations. Wav2Lip has provided its code for calculating FID and Sync_c and we use the same for these metrics. We use face-alignment [281] for landmark detection and the LMD metric proposed in [279]. For SSIM and PSNR we use the same inputs as used for FID, consequently, our values are a bit different compared to [14]. This is possibly because they evaluate these metrics at different scales and lack these evaluation details. On the other hand, we tend to get better LMD and Sync_c values for PC-AVS than noted in their paper. Some papers like [259] show PC-AVS's metrics only for reference as its generations occasionally fail in landmark detection. For uniformity of scale, we uncropped PC-AVS's generations and

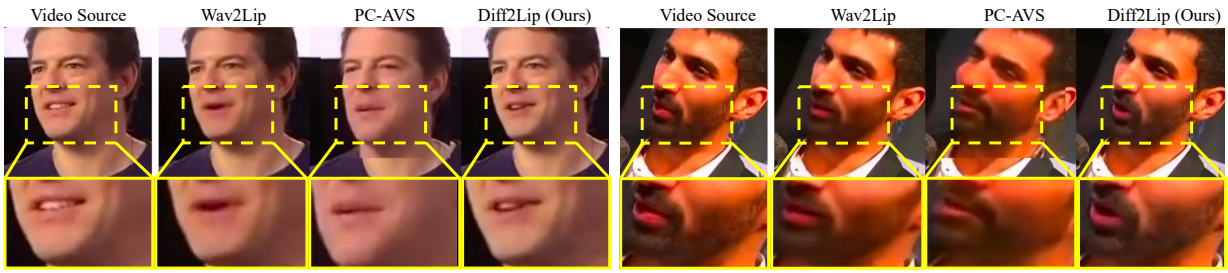


Figure 6.14: **Qualitative Visual-quality Comparison**. We zoom in on the mouth region of two examples and compare them against the video source. Wav2Lip [13] blurs the lip region while PC-AVS [14] tends to change the identity. Diff2Lip preserves identity and generates high-fidelity lips.

paste them back on the background before evaluation.

Reconstruction. Similar to the setting mentioned in [14] and later used in [259] and [262], we evaluate the methods on the task of reconstruction of the video given only the first frame and the audio corresponding to the same video. The audio is used as the driver for this reconstruction. Note that PCAVS requires an additional pose input apart from the identity input. We feed the first frame to it for both inputs, similar to the "fixed pose" setting in their paper. For Wav2lip and ours, the first frame acts as both the input frame and the reference frame. For Voxceleb2, we show the results on 4911 test utterances instead of 35K videos due to resource constraints. These 4911 utterances are the first utterance of each video in the test set, and hence cover all the videos in the test set. For LRW, our results are noted on all 25K videos in the test set.

The results are noted in Table 6.4. We observe that Diff2Lip outperforms both the methods with respect to FID metric on both datasets which points towards better generation quality. The SSIM, PSNR and LMD values of our method are comparable with the other methods. We see that Wav2Lip's Sync_c is better than both ours and PC-AVS's. This is possibly because our SyncNet expert may be weaker in performance compared to the one used in Wav2Lip.

Cross generation. We also evaluate the method on the task of lip-sync when the identity and the pose are controlled using a video while the lip-sync is driven using input audio corresponding to a

Table 6.6: User Study measured by Mean Opinion Scores (MOS) (max. 5) and Preference in percentage.

Measure	Wav2Lip	PC-AVS	Diff2Lip
MOS (Visual quality) ↑	3.75	2.71	**4.16**
MOS (Lip-sync quality) ↑	3.84	3.34	**3.86**
MOS (Overall quality) ↑	3.70	2.91	**3.91**
Preference ↑	37%	8.33%	**54.67**%

different video. This was introduced in [13] and is a more realistic setting as here the generations are closer to lip-sync in-the-wild. We use the input frame as the reference frame in this setting similar to Wav2Lip, as that provides the best texture information of the frame. For PC-AVS, the input frames are fed as both pose and identity sources. Note that we cannot evaluate SSIM and PSNR for this setting because there are no ground truth frames available. So, we provide the rest of the metrics in Table 6.4. For Voxceleb2, we select 4970 pairs of audio-video combinations where the two sources are different. We sample these using all the pair combinations of the first utterance of the first video coming from 71 randomly chosen test identities. For LRW, we use the 28K audio-video pair provided in Wav2Lip's [13] evaluation. The results are noted in Table 6.4. Similar to reconstruction evaluation, we here as well observe that our method excels in image quality while being comparable in other metrics except for Sync_c.

6.2.2.2 Qualitative Evaluation

For qualitative evaluation, we show visually compare against on both reconstructions (in Fig. 6.12) as well as cross generation (in Fig. 6.13). These settings are the same as introduced in Section 6.2.2.1. It can be observed in these qualitative results that PC-AVS tends to lose the identity of the source video and also suffers from boundary discontinuity problems which make it unsuitable for in-the-wild generation. On the other hand, Wav2Lip tends to generate blurred-out

mouth regions so as to achieve good lip sync. Diff2Lip does not suffer from these issues and is able to generate high-quality mouth region while having expressive lip shapes which correctly correspond to the ground truth (audio sources' mouth shape) as seen in Fig. 6.14.

User Study. We conduct a user study where we ask 15 participants to judge lip-sync videos generated in cross generation setting by Diff2Lip and two other methods. 20 videos were sampled from the VoxCeleb2's test set and are driven by randomly selected driving audios. The participants rated the videos 1-5 (where higher is better) in the aspects of (1) **Visual quality** (2) **Lip-sync quality**, and (3) **Overall quality**. We used the Mean Opinion Score (MOS) measure to aggregate these ratings. Further, we record the percentage of times users preferred a method. We present the results in Table 6.6, where we see that our method surpasses others in all the categories. In terms of Lip-sync quality, this is opposed to our quantitative results, especially Sync_c. We speculate that Sync_c, might favor blurry generations with high temporal consistency while humans prefer high fidelity over slight temporal inconsistency.

6.2.2.3 Ablations

We conduct an ablation study to showcase the contribution of various losses used during training. Specifically, we train our model in three different settings in which we introduce an additional loss in each setting. First, we train our model using only $\mathcal{L}_{\text{simple}}$ (Reconstruction). Second, we train another version of the model using $\mathcal{L}_{\text{simple}} + \mathcal{L}_2 + \mathcal{L}_{\text{sync}}$ (+ SyncNet). Here intuitively the $\mathcal{L}_{\text{sync}}$ should introduce better synchronization. Third, we further add a perceptual loss $\mathcal{L}_{\text{sync}}$ (+ Pecept). We add this loss because adding the SyncNet loss led to worse image quality. Finally, we add the sequential adversarial loss $\mathcal{L}_{\text{GAN}}$ to achieve even temporal consistency(+ Seq. GAN).

We test these on a smaller subset of 500 VoxCeleb2 test audio-video pairs in the cross generation setting as well as the reconstruction setting.

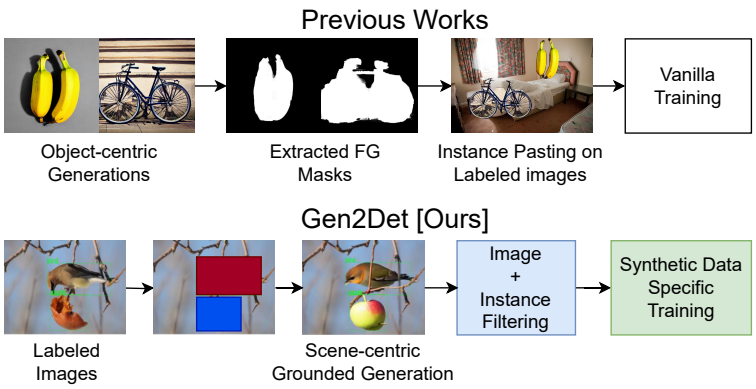
It can be seen in Table 6.5 that moving from "Reconstruction" to "+ SyncNet" gives a sudden improvement in the Sync_c metric. This supports our intuition that only reconstruction-based losses are not enough. We also see that this transition deteriorates the image quality. This gets solved as we move to the "+ Perceptual" setting. Finally, the addition of sequential adversarial loss not just further improves the image quality but also improves the Sync_c, clearly showing the advantage of this loss. In the reconstruction setting in Table 6.3, most of these observations still hold except FID being lower for "+ Perceptual" than "+ Seq. GAN". This could be attributed to the static nature of the input source in this setting while the ground truth is moving.

Part III: Generation for Recognition

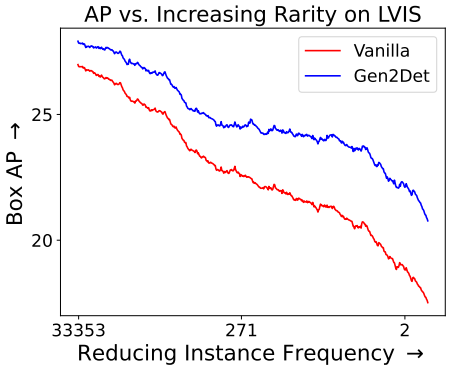
Chapter 7: Gen2Det: Generate to Detect

Recent developments in generative modeling using diffusion models has drastically improved the quality of the generation. Works like LDM [282], DALL-E [283, 284], Imagen [285], Parti [286] have shown the generation power which diffusion models possess. In addition to these models which can generate high quality images given a text input, there have been multiple developments in the direction of higher control in generation. Along this direction exist works which use conditional control like ControlNet [287] and GLIGEN [288]. There are also works like LoRA [289] and Dreambooth [290] which provide means to adapt these large models in a quick and efficient manner to generate specific kinds of images. With these developments in both quality and control of synthetically generated images, it is only natural to come back to the question of "How to best utilize data generated from these models for improving recognition performance?".

Most previous works [291, 292, 293] have explored using synthetic data from diffusion models for classification and pre-training tasks. The goal of our work is to look at utilizing more realistic scene configurations by generating images conditioned on the existing layout of boxes and labels and use them for object detection and segmentation. A recent work, XPaste [294], explores this problem and utilizes techniques similar to simple copy-paste [295] to paste synthetically generated object-centric instances onto real images for object detector training. However,



(a) Comparison with existing approaches.



(b) Larger improvements for rarer categories.

Figure 7.1: Existing approaches which utilize synthetic data for detection training (a [top]) follow a common methodology of generating object-centric images and pasting instances on real images. Gen2Det (a [bottom]) instead utilizes state-of-art grounded inpainting diffusion model to directly generate scene-centric images. Further, Gen2Det performs filtering to handle unsuitable generations both at image and instance level. Finally, during detector training we introduce changes to handle the filtered synthetic data better. As a result, our method consistently improves over vanilla training and the AP improvements increase (b) as the classes becomes rare (*i.e*, long-tailed classes).

their method relies on off-the-shelf segmentation methods built over CLIP [62, 296, 297, 298, 299] to extract masks, and is thus subject to segmentation errors. In addition to the extra components and compute, the generated images from XPaste are also not realistic as they do not respect natural layouts due to random pasting as shown in Figure 7.1a (top). In contrast, as shown in Figure 7.1a (bottom), Gen2Det leverages state-of-art diffusion models for grounded inpainting to generate scene-centric images which look more realistic. The goal of our approach is to show that utilizing such generated data from state-of-art diffusion models can lead to improvement in performance of object detection and segmentation models. By using a grounded inpainting diffusion model as our base generator, we are able to generate synthetic versions of common detection datasets like LVIS [300] and COCO [68] in a layout conditioned manner. We then carefully design a set of filtering and training strategies, with which we demonstrate that we can improve detection performance when training on the joint set of real and synthetic images. More

specifically, we sample batches of synthetic and real images with a sampling probability. During loss computation we also modify the loss for synthetic data to account for filtered instances. Additionally, without utilizing any additional models or data (including segmentation masks) during training, we show improvements in segmentation performance as a byproduct. The clear improvement over vanilla training shown in Figure 7.1b for classwise Box AP with increasing rarity of classes makes a strong case for such a pipeline especially in long-tailed or low data regimes.

We emphasize the importance of the proposed filtering and training techniques to incorporate synthetic data through our ablations, as we show that directly training on the generated data or mixing it with real data in a naive manner ends up hurting the performance. It is important to do the proper filtering and loss modifications to mitigate shortcomings of the generations and let the model learn from the mix of real and synthetic data. We propose Gen2Det as a general approach to utilize synthetic data for detector training. Due to its modular nature, different components can be updated as the field progresses. This includes utilizing better generators, filtering techniques, architectures and training recipes. We highlight our contributions below:

- Propose to use state-of-art grounded-inpainting models to generate synthetic data in a manner that respects the realistic scene layouts.
- Propose techniques to perform filtering at image and instance level along with changes in the detector training in order to utilize such synthetic data effectively and handle imperfections in generated instances.
- Through our experiments we show consistent improvement across datasets and architectures especially in long-tailed and low data settings. There is also considerable gains for

rare categories which are shown on the long tailed LVIS dataset.

- We also provide quantitative and qualitative ablations to showcase the effect of different hyperparameters and components of our approach which would help further research in usage of such synthetically generated data.

7.1 Related Works

Diffusion Models for Controllable Image Generation. Diffusion models have been shown to generate images with unprecedented high quality. Amongst these models a few of the popular models which can generate images from text prompts include LDM [282], DALL-E [283, 284], Imagen [285], and Parti [286]. In addition to just text guided generation there have been further improvements in having more controllable generation and editing through different inputs [287, 288, 301, 302, 303, 304, 305, 306]. One of the popular works ControlNet [287] provides multiple ways to enable control including using edge maps, scribbles, segmentation masks amongst other modalities. GeoDiffusion [307] and LRDiff [308] also provide novel ways to condition and generate grounded images. Another work GLIGEN [288] also performs grounded generation and inpainting conditioned on boxes, keypoints, HED maps, edge maps and semantic maps with an open vocabulary.

Synthetic Data for Detection. There has been prior non-diffusion based work exploring the use of synthetic data for detection. Some works have explored using image-based blending all the way to depth and semantics informed positioning [309] for creating synthetic data. There have been other works which use computer graphics based approaches to render synthetic data by varying lighting and orientations [310, 311] to generate variations in instances. Further there

have been a line of augmentations belonging to the copy-paste family [295, 312, 313, 314] where object instances are cropped and pasted on images at different locations. All these approaches end up repeating instances already present in the training set.

Leveraging Synthetic Data from Diffusion Models. Due to its high quality generation and the flexibility in handling multimodal data (*e.g*, vision and language), there has been recent work in using pretrained diffusion models for different tasks. Peekaboo [315] shows the use of diffusion models as zero shot segmentors while other works [316, 317] have shown that diffusion models can act as zero shot classifiers. StableRep [293] uses data generated by diffusion models to train self-supervised models. Some works explore using diffusion model features [318, 319, 320, 321, 322] directly for downstream tasks. There have also been works exploring the use of synthetic data from diffusion models to improve image classification [291, 292, 323, 324, 325]. While the classification task has had a lot of exploration with synthetic data due to the object-centric nature of images which these models can generate easily, the detection task is less explored as it's harder to generate data with grounded annotations using these models. Recent studies [326, 327, 328] explored the use of such data in constrained few shot settings for detection where the gains are expected. Another work FreeMask [329] utilizes synthetic data for panoptic segmentation but requires segmentation masks to generate data. Recently, XPaste [294] showed consistent improvements using the CenterNet2 [330] architecture. Specifically, XPaste [294] uses diffusion models to generate object-centric images and uses multiple CLIP [62] based approaches [296, 297, 298, 299] to extract segmentation maps which makes it slow and error-prone. Following the extraction of instances and their segmentation maps, they use synthetic and real instances (retrieved from an additional real dataset) to perform copy-paste [295] augmentation to generate synthetic data. Gen2Det on the other hand does not use any additional CLIP based approaches.

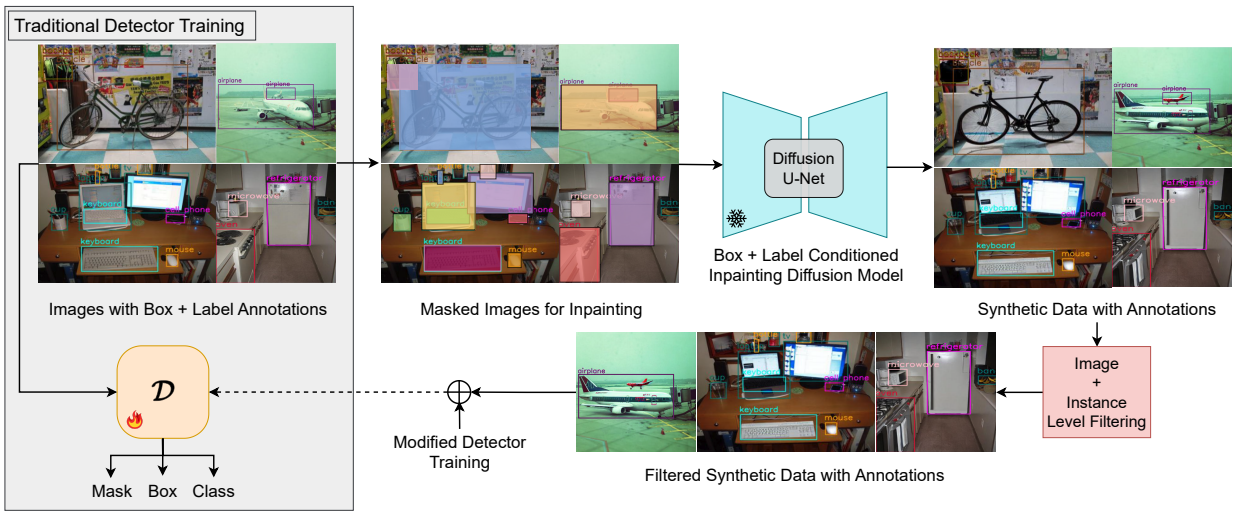


Figure 7.2: **Gen2Det: our proposed pipeline for generating and utilizing synthetic data for object detection and segmentation.** Gen2Det starts by generating grounded inpainted images using state-of-art diffusion model. The generated images are then filtered at image and instance level to remove globally bad images as well as low quality individual instances. Finally, we train object detection and segmentation models using the filtered data along with our improved training methodology by introducing sampling and background ignore.

Rather, once we generate the data we only train with bounding box labels but show improvements in both box and mask AP. Also in terms of training speed, we are $3.4\times$ faster compared to XPaste with the same configuration.

7.2 Approach

7.2.1 Overview

Through Gen2Det, we provide a modular way to generate and utilize synthetic data for object detection and instance segmentation. As shown in Figure 7.2, we start by generating data using state-of-art grounded image inpainting diffusion model. As the generations may not always be perfect we perform image level and instance level filtering. The image level filtering removes overall poor quality synthetic images while the instance specific filtering discards poor

quality instance annotations. The image level filtering is performed using a pre-trained aesthetic classifier [331] while the instance level filtering is performed using a detector trained on the corresponding real data. After performing filtering, the images are suitable for training using our proposed training strategy. While we do not make any architectural changes to the detector to keep the pipeline general enough, we perform a probability based batch sampling to select the data source between synthetic and real images. Further while computing the losses, we modify the negatives corresponding to the synthetic data to be ignored from loss computation to account for the filtering we performed.

7.2.2 Image Generation

We start our pipeline by generating synthetic images by utilizing a state-of-art grounded inpainting diffusion model. This model is trained to support multiple kinds of input conditions. We utilize the model trained for image inpainting with the image, boxes, and corresponding labels as input. Specifically, for each image $\mathcal{I}$ in the dataset with box annotations $\mathcal{B}$, we provide as input the image and the corresponding annotations to the diffusion model and ask it to inpaint the annotated regions with new instances of the same class by providing the box label as input. As inpainting model requires an image and box level text description, for each box b_i we use the class name $<c_i>$ as the box level prompt and a concatenation of strings $<$a c_1, a c_2, ... and a $c_n>$ as the image level prompt where n is the number of instances in the image. We show examples of images generated using this technique in Figure 7.3.

7.2.3 Filtering

The images generated using the strategy described above may not always contain good generations so it is important to filter out low quality generations before using them to train a detection model. We perform two levels of filtering. First we filter at image level followed by a more granular instance level filtering. We describe each of these below.

Image Level Filtering. Image level filtering removes synthetic images that appear visually unappealing at a global scale. We do this by utilizing a pretrained model [331] which is trained to predict how pleasing the image looks visually by assigning each image an aesthetic score. A higher score indicates a more aesthetically pleasing image. Based on qualitatively analyzing images and computing the average aesthetic score for real images (from COCO) we set a threshold for aesthetic filtering τ_a. Any image with an aesthetic score less than τ_a is discarded and its corresponding annotations are removed. We show the effect of this filtering by visualizing discarded samples in Figure 7.4.

Instance Level Filtering. As a more granular level of quality assurance, we also perform an extra filtering step at the instance level. This filtering is designed to remove annotations for specific generated instances which do not have good generation quality. In order to perform this filtering we first train a detector on only the real data. We then pass all the generated images through the trained detector and store its predictions. Based on the detectors predictions we evaluate whether a ground truth annotation corresponding to an inpainted region should be utilized for training or not. This step is important to handle poor quality/incorrect generations. To do this, for each generated image we iterate over all ground truth annotations used to generate it. For each annotation we go through all predictions and remove the ground truth annotation if there

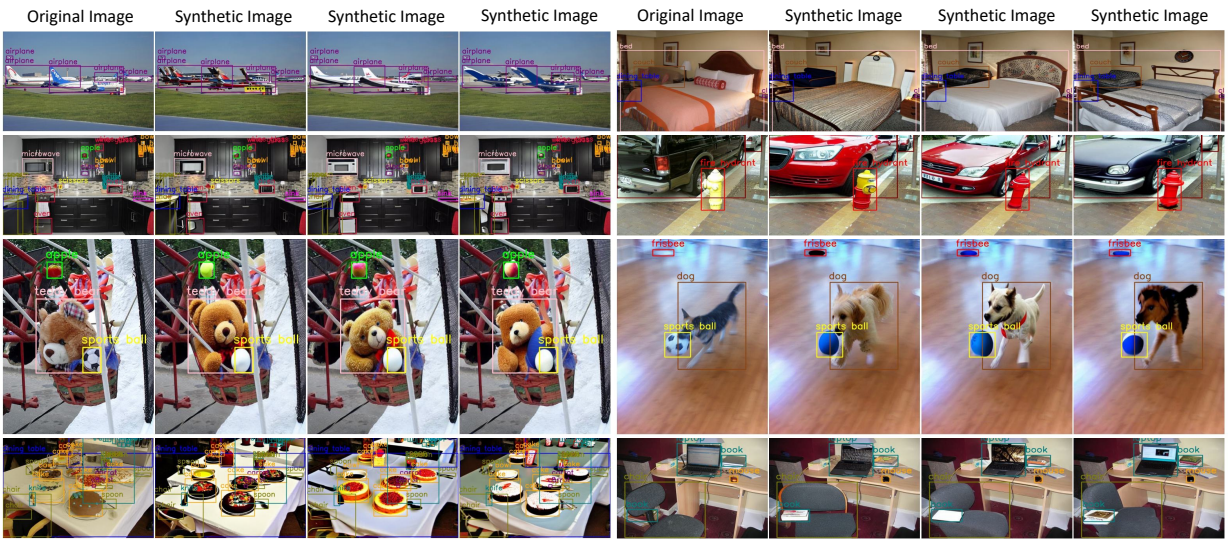


Figure 7.3: Examples of synthetically generated data using COCO. The first and fifth columns correspond to the original COCO images and the rest of the columns are generations with different seeds.

is no overlapping prediction with a score greater than τ_s and IoU greater than τ_{iou}. This helps remove ground truth annotations corresponding to generation which are either bad quality as the pre-trained detector is not able to predict it with even a low confidence threshold, or for which the region where they are generated is not the ground truth region. We show results of the kind of annotations removed using this filtering in Figure 7.5.

7.2.4 Model Training

Once the data is ready for use we describe our training strategy to utilize the synthetic dataset along with the real dataset. It should be noted that for synthetically generated images we do not have the segmentation masks for the generated instances so we do not apply the mask loss for those images.

Batch Sampling. We utilize both the real and synthetic data for training the final model. To do this, we need to figure out how to mix the two datasets together in a way which lets us utilize the full potential of the real data but at the same time extracts as much useful information

from the additional synthetic data as possible. Naively using only the synthetic dataset and a naive combination of synthetic and real datasets during training leads to drop in performance as shown in the ablations. Therefore, to best use both the real and synthetic datasets, we end up defining a sampling probability p. Specifically, a batch is chosen to comprise of completely synthetically generated samples with the sampling probability p during training. This sampling strategy interleaves batches of synthetic and real data and as we will show leads to effective learning from both data sources.

Background Ignore. While the filtering we perform at both image and instance level can deal with bad quality generations, they also introduce some noise especially for classes for which the detector itself has poor quality predictions. Essentially, the instance level filtering removes ground truth annotation corresponding to bad quality/incorrect generations but that does not remove the bad instance itself from the image. Additionally, the generative model could have also hallucinated multiple instances of an object class leading to missing annotations. To counter the effect of both these scenarios we introduce an ignore functionality during training. Essentially, for both the region proposal network (RPN) and detector head we ignore the background regions from the loss computation, if their fg/non-bg class prediction score is higher than a threshold τ_i. This lets us train even in the presence of some bad quality regions without incorrectly penalizing the detector for predicting objects at those locations. This is an important change in training to allow our model to effectively utilize synthetic images with incorrectly hallucinated instances, as the instance level filtering only removes the instance annotation but does not handle hallucinations.

Additionally, as stated before we do not utilize any additional mask information for the synthetic data as we only inpaint using boxes. Due to the lack of masks for synthetic data we

Table 7.1: Comparisons on LVIS using Centernet2 architecture. We report the overall AP and also AP for rare classes (AP_r) for both box and mask evaluations.

Method	Box		Mask	
	AP	AP_r	AP	AP_r
Vanilla (real only)	33.80	20.84	29.98	18.36
Vanilla (synth only)	11.27	6.54	-	-
Vanilla (synth+real)	31.82	20.68	27.49	18.35
XPaste	34.34	21.05	30.18	18.77
Ours	**34.70**	**23.78**	**30.82**	**21.24**

Table 7.2: Comparisons with baselines on COCO dataset using Centernet2. We report the Box AP (AP_b) and Mask AP (AP_m).

Method	AP_b	AP_m
Vanilla (real only)	46.00	39.8
Vanilla (synth only)	24.51	-
Vanilla (synth+real)	44.35	37.03
XPaste	46.60	39.90
Ours	**47.18**	**40.44**

ignore the mask loss for the synthetic images. Even with this, as shown in our quantitative results, we see an improvement in both box and mask performance.

7.3 Results

7.3.1 Experimental Setting

We evaluate our approach on LVIS [300] and COCO [68] datasets. These datasets provide a standard benchmark for object detection and instance segmentation approaches. LVIS is a long tailed dataset with 1203 classes utilizing the same images as COCO which contains only 80 classes. The long tailed nature of LVIS makes it especially appealing to showcase the use of synthetic data for the rare categories. We report the Average Precision (AP) for both these datasets and compare using the box AP (AP_b) as well as mask AP (AP_m). For LVIS based on the dataset definition we also report the AP for rare categories separately to highlight the improvements there. Further, we also report results on artificially created subsets of COCO to simulate the low-data regime by using 1%, 5%, 10%, 20%, 30%, 40% and 50% randomly selected images from COCO.

We utilize the Detectron2 framework [134] for training all the models and use the XPaste code to run their baselines. We use the Centernet2 clone adapted for instance segmentation. For LVIS, we rerun XPaste to get the results corresponding to adding only additional synthetic data. For COCO, we directly adopt their evaluation numbers due to the lack of corresponding configs and details for reproduction. Following their code we also utilize the ImageNet-22k [93] pretrained models for the Centernet2 experiments while using the ImageNet-1k [93] models from Detectron2 for the rest of the experiments which is common practice. We utilize the Mask R-CNN [332] architecture and the LVIS dataset for our ablations. The Mask R-CNN [332] and Faster R-CNN [116] are implemented in the Detectron2 framework [134].

For data generation, we utilize a state-of-art diffusion model for grounded inpainting using the images from LVIS and COCO along with their annotations which contain the boxes and class labels. For aesthetic filtering, we utilize the open source model available in their repository [331]. For aesthetic filtering we set τ_a to 4.5 which is roughly the same as the average aesthetic score on real COCO images. For instance level filtering we set τ_s as 0.2 and τ_{iou} as 0.3 for LVIS. We use the same τ_{iou} for COCO and set τ_s as 0.1. During training, we use a sampling probability $p = 0.2$. We set τ_i to 0 for both datasets thus effectively ignore all background regions corresponding to the synthetic data from the loss.

On 8 GPUs generation takes $\sim$1.25s per image generation. $1\times$ synthetic LVIS data contains 100,170 images and 1,270,141 instances. Image level filtering reduces the images and instances to 77,513 and 1,005,854 respectively. Instance level filtering reduces the instances to 472,114 with the same image count as it does not remove any images. These are fairly consistent across generation seeds with similar ratios of removal for COCO.

Table 7.3: Results across different backbones on the LVIS dataset. We report both Box and Mask AP overall as well as for the rare (AP_r) categories.

Method	Box		Mask	
	AP	AP_r	AP	AP_r
Faster R-CNN	21.39	10.05	-	-
Faster R-CNN (Ours)	**22.90**	**12.78**	-	-
Mask R-CNN	22.29	10.63	21.83	11.15
Mask R-CNN (Ours)	**24.42**	**15.43**	**23.67**	**15.33**
Centernet2	33.80	20.84	29.98	18.36
Centernet2 (Ours)	**34.70**	**23.78**	**30.82**	**21.24**

Table 7.4: Comparisons across architectures on COCO dataset. We report the Box AP (AP_b) and Mask AP (AP_m).

Method	AP_b	AP_m
Faster R-CNN	40.20	-
Faster R-CNN (Ours)	**40.56**	-
Mask R-CNN	41.08	37.14
Mask R-CNN (Ours)	**41.53**	**37.46**
CenterNet2	46.00	39.8
CenterNet2 (Ours)	**47.18**	**40.44**

7.3.2 Quantitative Results

Comparison on LVIS. We start by comparing our approach to the existing works in Table 7.1 on the LVIS dataset with CenterNet2 backbone. Over vanilla training we show that our method improves the Box and Mask AP over rare categories by 2.94 and 2.88 respectively with a 0.9 and 0.84 Box and Mask AP improvement overall. Despite the fact that XPaste utilizes four different CLIP based approaches [296, 297, 298, 299] to obtain segmentation masks and is also $3.5\times$ slower to train compared to our approach on LVIS, we are able to outperform XPaste by 2.73 Box AP and 2.47 Mask AP on the rare categories and by 0.36 and 0.64 Box and Mask AP across all categories. The huge gains in rare categories and overall improvements highlight our methods effectiveness in both long tailed as well as general settings.

Comparison on COCO. We compare on the COCO benchmark in Table 7.2. On COCO too we show an improvement of 1.18 and 0.64 on Box and Mask AP over vanilla training. Compared to XPaste we improve by 0.58 Box AP and 0.54 Mask AP. We note that compared to LVIS the improvements are slightly lower here as LVIS is a more long-tailed dataset where adding synthetic data shines.

Table 7.5: Performance on the LD-COCO setting. The LD-COCO data is a randomly selected subset of COCO with different percentages of images from the dataset. We report results using both Box AP (AP_b) and Mask AP (AP_b).

Method	AP	Image %							
		1	2	5	10	20	30	40	50
Vanilla	AP_b	11.76	16.07	19.52	25.29	29.32	33.02	34.39	36.52
	AP_m	11.34	15.36	18.48	23.35	26.97	30.37	31.36	33.20
Gen2Det	AP_b	14.84 (+3.08)	18.42 (+2.35)	22.91 (+3.38)	27.62 (+2.33)	31.76 (+2.44)	34.70 (+1.68)	36.09 (+1.69)	37.74 (+1.22)
	AP_m	13.89 (+2.55)	17.27 (+1.91)	21.17 (+2.69)	25.34 (+1.99)	29.03 (+2.06)	31.65 (+1.28)	32.67 (+1.30)	34.24 (+1.04)

Comparison on LD-COCO. In order to simulate a low-data regime on COCO with fewer images and labels, we create random subsets of different percentages of COCO images. In Table 7.5 we show results on the low-data (LD) version of COCO. We show consistent improvement in both Box and Mask AP across different image percentages. On average we improve the Box and Mask AP by 2.27 and 1.85 points respectively. This shows that adding synthetic data is especially fruitful in both long tailed and low data regimes.

Comparison Across Backbones. We further show comparison across architectures by comparing to performance with vanilla training using only the real data. Table 7.3 shows comparison on LVIS dataset. We observe consistent improvement in overall box and mask AP, and as mentioned above, our method especially shines for the rare categories. For the MaskRCNN architecture we show a substantial gain in performance with an improvement of 2.13 Box AP and 1.84 Mask AP over just training on real data. The improvements on rare categories are even higher with a gain of 4.8 Box AP and 4.18 Mask AP, highlighting the efficacy of using synthetic data in the Gen2Det pipeline for rare/long-tailed categories. Table 7.4 shows a similar analysis on COCO. For COCO too we see consistent improvements in both Box and Mask AP across different architectures.

7.3.3 Ablation

Effect of Different Components. Table 7.6 summarizes the effect of incorporating different components in our pipeline. We report the Box AP and Box AP_r for this analysis along with the respective improvement deltas introduced by each component over the baseline. Row 1 and 2 correspond to only training with the real and synthetic data respectively. It can be noted that naively training on only the synthetic or a simple combination of real and synthetic data (row 3) leads to a degradation in overall AP. Combining the two sets (row 3) does lead to a marginal increase in Box AP_r due to the addition of more instances of the rare categories. Adding synthetic data along with sampling (row 4) gives us a boost of 1.2 in overall Box AP and 2.58 for the rare categories.

Further, incorporating image level filtering which removes bad quality images in their entirety gives us a improvement of 1.49 Box AP and 3.16 Box AP_r (row 5). Following our filtering pipeline, performing instance level filtering on its own (row 6) does not work much better than just image level filtering as it ends up removing annotations corresponding to bad quality generations but the instances still remain in the images. Additionally, we also try performing background loss ignore for the synthetic data without instance level filtering (row 7), and that does not lead to an improvement on its own rather leads to a drop in performance. Finally, in row 8 we see that instance level filtering coupled with background ignore takes care of the discarded annotations and leads to a net total improvement of 2.13 AP and 4.8 AP_r. It should be noted that instance level filtering alone only removes certain annotations from training, using it with background ignore unlocks the full potential of such filtering and is important for training with synthetic data which might not always be perfect.

Table 7.6: Ablation of various components of proposed approach on LVIS using Mask R-CNN. We report the Box AP overall as well as rare (AP_r) categories.

Row	Real Data	Synth. Data	Sampling	Img. Filt.	Inst. Filt.	Bg. Ignore	Box AP	Δ_{AP}	Box AP_r	Δ_{AP_r}
1	✓	✗	✗	✗	✗	✗	22.29	–	10.63	–
2	✗	✓	✗	✗	✗	✗	7.67	-15.25	4.57	-6.06
3	✓	✓	✗	✗	✗	✗	20.75	-1.54	10.91	0.28
4	✓	✓	✓	✗	✗	✗	23.49	1.2	13.21	2.58
5	✓	✓	✓	✓	✗	✗	23.78	1.49	13.79	3.16
6	✓	✓	✓	✓	✓	✗	23.76	1.47	13.62	2.99
7	✓	✓	✓	✓	✗	✓	23.32	1.03	12.77	2.14
8	✓	✓	✓	✓	✓	✓	**24.42**	**2.13**	**15.43**	**4.8**

Table 7.7: Ablation on different hyperparameters on the LVIS dataset with Mask R-CNN backbone.

Config	1×	2×	3×	4×
AP	22.50	**24.42**	22.85	22.31

(a) Longer training schedule.

Probability	0.0	0.1	0.2	0.3	0.4	0.5
AP	22.29	23.23	**24.42**	23.42	23.28	22.52

(b) Effect of sampling probability.

Synthetic Data	0×	1×	2×	3×	4×	5×
AP	22.29	23.55	**24.42**	23.67	23.82	23.82

(c) Effect of generated data sample counts.

Effect of Longer Training. We try to see the effect of longer training by increasing the number of iteration for the detector training. We also adjust the learning rate steps proportionally for each experiment. As can be seen from Table 7.7a, starting with the $1\times$ config and moving to $2\times$ gives us a boost as it is able to utilize the additional samples better but further longer training does not work that well reduces performance as it starts overfitting to the data.

Effect of Sampling Probability. Here we vary the sampling probability p used during training which affects whether the batch should be made from real or synthetic data. We show the change in performance for different synthetic probabilities in Table 7.7b. As can be seen in all cases we are better than the vanilla training. We do see that initially increasing the sampling

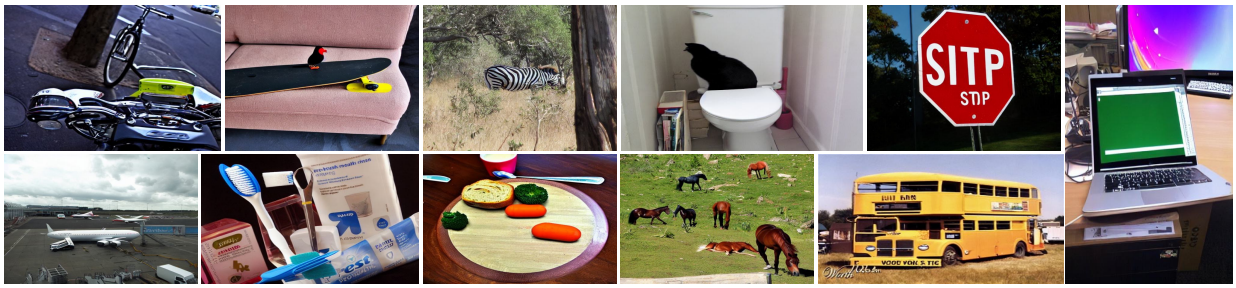


Figure 7.4: Examples of samples discarded during the image level filtering. As can be seen, image level filtering is able to remove images with artifacts present at a global level.

probability to 0.2 helps and improves performance after which it decreases as the probability is further increased. This might be because the real data which is pristine is seen lesser and lesser after a certain probability value, and therefore we might get negatively impacted by the noise in synthetic data.

Effect of More Synthetic Samples. Similar to longer training, adding more synthetic samples also has a curved performance as shown in Table 7.7c. Adding $1\times$ data is better than training on only real data. Further increasing the number of synthetic samples to $2\times$ improves the performance. In our setting though, increasing them more leads to a reduction in performance. Ideally, the goal would be to keep increasing the performance with more synthetic data but we leave that for future work as it would require looking into improving diversity of generations which will happen as the generators improve. Also adding more data under the current scheme ends up generating more images with the same layout leading to a drop after a certain point. Increasing diversity in layouts would be another way to improving performance further. Again, it should be noted that in all cases, our performance is sufficiently better than vanilla training.

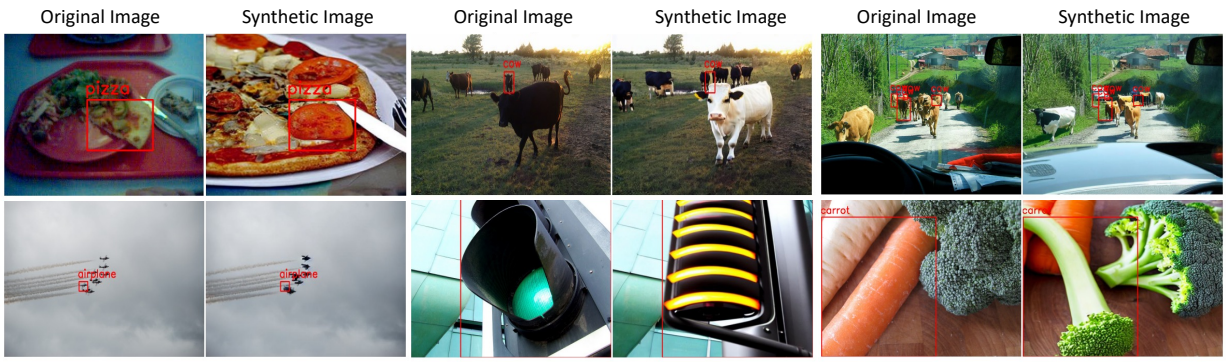


Figure 7.5: Examples of ground-truth instance annotations discarded by the instance level filtering (red box). Instance level filtering is able to remove poor quality, missing, or incorrect generations.

7.3.4 Qualitative Results

We show qualitative outputs of different parts of pipeline using the COCO dataset. In Figure 7.3 we show a few samples which are the scene-centric synthetic images generated using the grounded inpainting model. The first column corresponds to the real image from COCO dataset followed by multiple synthetic generations each with a different seed leading to diverse instances.

We highlight a few samples which are discarded from the image level filtering step in Figure 7.4. As can be seen these are images which do not look good at an image level even though some of the instances might look fine. For example in the first image in first row, the bicycle generation in the front seems incorrect. Similarly, in the first image in the second row, the bigger airplane which covers a major part of the image does not look realistic.

In Figure 7.5 we show some examples of instances discarded by instance level filtering. It can be seen that instance level filtering discards instances which have either been incorrectly generated or are bad quality generations. For example in the first example in the first row, the generation does not correspond to the pizza rather a specific topping. Similarly, in the first example

in the second row, there is no airplane generated in the ground truth region.

7.4 Discussion

With the huge strides in image generation in terms of both quality and control, we try to tackle the problem of training detection and segmentation models with synthetic data. Our proposed pipeline Gen2Det utilizes state-of-art grounded inpainting diffusion model to generate synthetic images which we further filter at both image and instance level before using in training. We also introduce some changes in the detector training to utilize the data better and take care of shortcoming which the data might pose both in terms of poor quality, incorrect or missing generations. Along with detailed quantitative ablations and qualitative outputs we show the efficacy of different components of our approach. Finally, we show improvement across both LVIS and COCO and show higher improvements on rare classes in the long tailed LVIS setting. Additionally, we show improvement in low-data COCO setting too. Most interestingly, we show improvement in segmentation performance without using any additional segmentation masks like existing works. We hope Gen2Det acts as a general modular framework which can benefit from future developments in both generation as well as detector training due to easily replaceable general purpose blocks.

7.5 Appendix

7.5.1 Experimental Details

For learning rate for Faster and Mask R-CNN experiments we use the default LR of 0.08 which is scaled from 0.02 as we use $4\times$ the batch size as the default configs. For LVIS, this

LR was not stable so we use an LR of 0.04 accross all Faster and Mask R-CNN experiments. For LD-COCO we do not perform filtering or background ignore due to already low number of images and instances. As for compute as mentioned in the main paper, on 8 A100 GPUs GLIGEN takes $\sim$1.25s per image generation. $1\times$ synthetic LVIS data contains 100,170 images and 1,270,141 instances. In total for both COCO and LVIS generating $1\times$ data takes around 1.5 days on one 8 A100 node. We generated data across multiple GPUs paralelly giving us flexibility to generate data much faster. For LVIS and COCO we generated $2\times$ and $4\times$ data respectively. As for detector training based on the configs the LVIS and COCO training takes 14.5 hrs and 17 hrs respectively on a single 8 A100 GPU node.

We also show quantitative results for the common and frequent categories for the LVIS dataset. Table 7.8 shows the comparison iwth baselines while Table 7.9 shows comparison in performance across backbones.

Table 7.8: Comparisons on LVIS with baselines using Centernet2 architecture. We report the overall AP and also AP for rare (AP_r), common (AP_c) and frequent (AP_f) classes for both box and mask evaluations.

Method	Box				Mask			
	AP	AP_r	AP_c	AP_f	AP	AP_r	AP_c	AP_f
Vanilla (real only)	33.80	20.84	32.84	40.58	29.98	18.36	29.64	<u>35.46</u>
Vanilla (synth only)	11.27	6.54	9.35	15.51	-	-	-	-
Vanilla (synth+real)	31.82	20.68	31.18	37.42	27.49	18.35	27.08	31.96
XPaste	<u>34.34</u>	<u>21.05</u>	**33.86**	**40.71**	<u>30.18</u>	<u>18.77</u>	<u>30.11</u>	35.28
Ours	**34.70**	**23.78**	<u>33.71</u>	<u>40.61</u>	**30.82**	**21.24**	**30.32**	**35.59**

7.5.2 Quantitative Results on COCO-O

Our work uses the commonly used COCO and LVIS benchmarks for evaluation which are widely-accepted and representative benchmarks. To evaluate the robustness perspective of using

Table 7.9: Results across different backbones on the LVIS dataset. We can see consistent improvement across various backbones with especially higher gains on the rare categories. We report both Box and Mask AP overall as well as for the rare (AP_r), common (AP_c) and frequent (AP_f) categories.

Method	Box				Mask			
	AP	AP_r	AP_c	AP_f	AP	AP_r	AP_c	AP_f
Faster R-CNN	21.39	10.05	19.46	28.51	-	-	-	-
Faster R-CNN (Ours)	**22.90**	**12.78**	**21.19**	**29.27**	-	-	-	-
Mask R-CNN	22.29	10.63	20.15	29.80	21.83	11.15	20.42	28.10
Mask R-CNN (Ours)	**24.42**	**15.43**	**22.63**	**30.38**	**23.67**	**15.33**	**22.62**	**28.51**
Centernet2	33.80	20.84	32.84	40.58	29.98	18.36	29.64	35.46
Centernet2 (Ours)	**34.70**	**23.78**	**33.71**	**40.61**	**30.82**	**21.24**	**30.32**	**35.59**

synthetic data fro training, we also evaluate our method on COCO-O [333] and achieve mean AP of **19.04** improving over vanilla training performance of 18.13 without any specific generations corresponding to cartoon, sketch etc. Additionally, the ablations show filtering improves performance, indicating the poor quality of filtered samples.

7.5.3 Pasting Object Centric Instances

We use state-of-art text to image diffusion model to generate object centric images for each instance annotation and paste them on corresponding LVIS images in a layout satisfying manner using the box coordinates. We use the prompt a photo of $< c >$ where c is the class name for that instance. We then train the detector with vanilla training and sampling synthetic and real data with equal probabilities. This gives us a Box AP of 22.50 and Mask AP of 21.45 which is lower than the performance obtained through Gen2Det where we get 24.42 Box AP and 23.67 Mask AP.

7.5.4 Ablations

7.5.4.1 Ablation on τ_s

We vary τ_s which is the score threshold we use for detector filtering. For this experiment we keep τ_{iou} fixed at 0.3. We show the results for this in Table 7.10. It can be seen that initially increasing the score threshold helps but after 0.2 increasing it further leads to a reduction in performance. This might be because we filter out too many annotations.

7.5.4.2 Ablation on τ_{iou}

We vary τ_{iou} which is the score threshold we use for detector filtering. For this experiment we keep τ_s fixed at 0.2. We show the results for this in Table 7.11. It can be seen that initially increasing the IoU threshold helps but after 0.3 increasing it further leads to a reduction in performance. This might be because we filter out too many annotations.

7.5.4.3 Ablation on τ_i

We vary τ_i which is the score threshold we use for background ignore. We show the results for this in Table 7.12. It can be seen that initially setting the score threshold to 0 which ends up ignoring all background instances from the synthetic data works the best. After that increasing it reduces it with sporadic increases at certain thresholds.

7.5.5 Qualitative Results

7.5.5.1 Qualitative Inpainted Generation

We show qualitative generation on the LVIS dataset in Figures 7.6 and 7.7. These generations utilize the LVIS annotations and images along with state-of-the-art grounded inpainting model to generate the instances conditioned on the class labels, box labels, and input images.

We also show qualitative generation on the COCO dataset in Figures 7.8 and 7.9 utilizing COCO annotations and images to generate them.

7.5.5.2 Qualitative Results for Image Filtering

In Figure 7.10 we show results of the image level filtering by visualizing images discarded during this step on the LVIS dataset. It can be seen there are high level issues in the generations. In the third image in row 1 the sign seems to have some weird patterns. In row 3 the second image has low quality aircraft generations. Similarly in the second image in the last row the cat generation is not great.

We also show more image filtering results on the COCO dataset and visualize the discarded images in Figure 7.11.

7.5.5.3 Qualitative Results for Detector Filtering

We present some examples for detector filtering on LVIS in Figure 7.12. We show the original and generated instances for each of the images and show the ground truth instance annotation(s) being discarded for each image in red. We also show examples of detector filtering on

Table 7.10: Comparisons across different τ_s keeping τ_{iou} fixed at 0.3 on the LVIS dataset. We report the overall AP and also AP for rare (AP_r), common (AP_c) and frequent (AP_f) classes for both box and mask evaluations.

τ_s	Box				Mask			
	AP	AP_r	AP_c	AP_f	AP	AP_r	AP_c	AP_f
0.1	23.85	14.04	21.89	30.36	23.16	14.21	21.92	28.49
0.2	**24.42**	**15.43**	**22.63**	**30.38**	**23.67**	**15.33**	**22.62**	**28.51**
0.3	23.59	13.21	21.74	30.21	22.89	13.50	21.66	28.39
0.4	23.75	13.48	21.97	30.25	23.03	13.81	21.86	28.38
0.5	23.62	13.69	21.56	30.28	23.05	14.24	21.75	28.36

Table 7.11: Comparisons across different τ_{iou} keeping τ_s fixed at 0.2 on the LVIS dataset. We report the overall AP and also AP for rare (AP_r), common (AP_c) and frequent (AP_f) classes for both box and mask evaluations.

τ_{iou}	Box				Mask			
	AP	AP_r	AP_c	AP_f	AP	AP_r	AP_c	AP_f
0.1	23.49	12.90	21.70	30.15	22.82	12.90	21.77	28.35
0.2	23.56	12.32	22.05	30.18	22.88	12.58	22.08	28.31
0.3	**24.42**	**15.43**	**22.63**	**30.38**	**23.67**	**15.33**	**22.62**	**28.51**
0.4	23.73	13.09	22.02	30.32	23.12	13.40	22.13	28.49
0.5	23.77	13.22	21.89	30.51	22.91	13.33	21.58	28.60

COCO in Figure 7.13.

7.5.5.4 Predictions on Generated Images

We visualize a few prediction on generated LVIS images in Figure 7.14.

7.5.5.5 Images Filtered at Different Thresholds

We visualize a few examples which are filtered at different image level filtering thresholds in Figure 7.15. We can observe from left to right higher score corresponds to better visual quality.

Figure 7.6: Examples of generations using the inpainting diffusion model on the LVIS dataset. The first column corresponds to the original LVIS images and the rest of the columns are generations with different seeds. These generated images are then fed to our filtering pipeline for further processing before use in training.

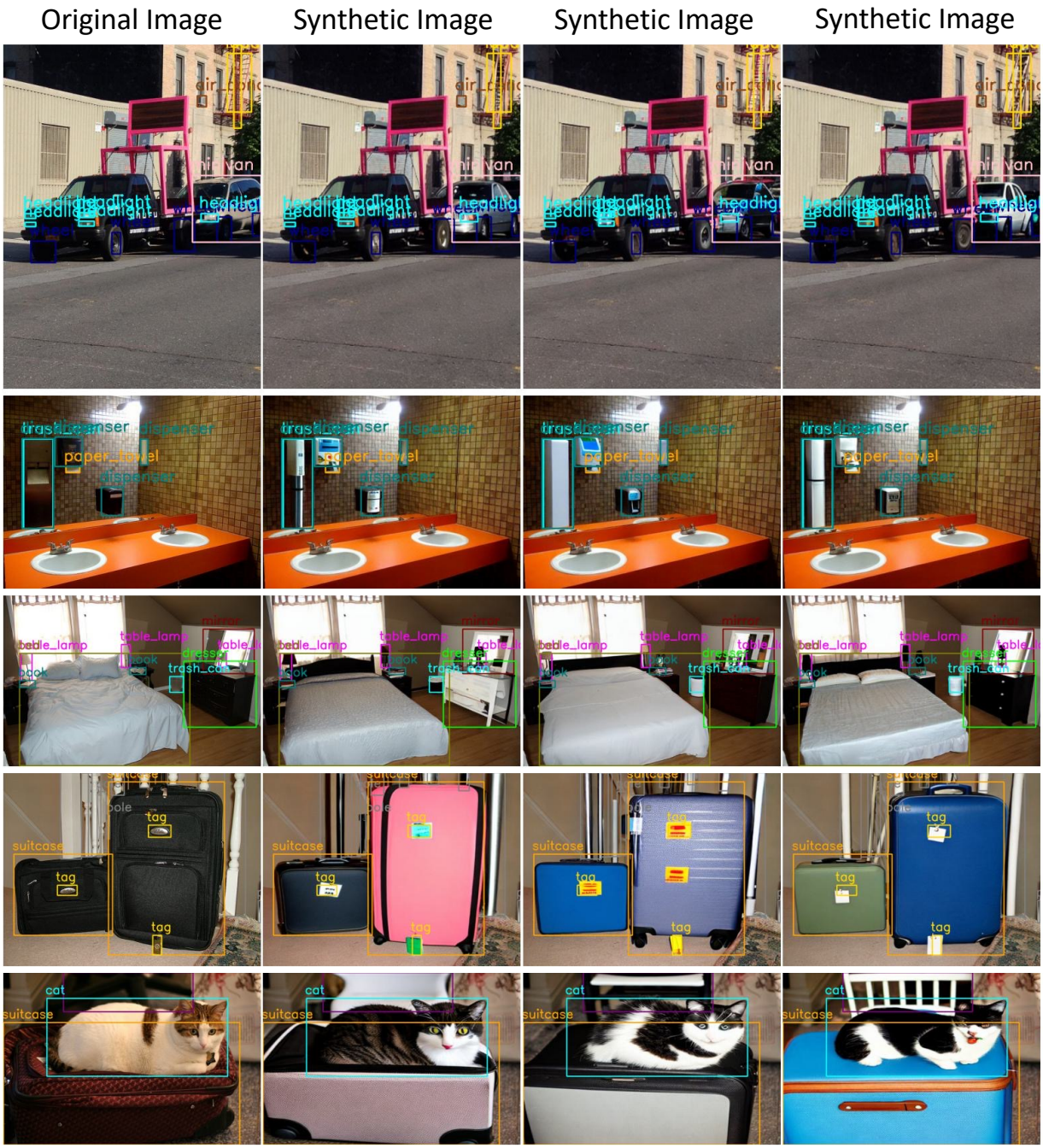


Figure 7.7: Examples of generations using the inpainting diffusion model on the LVIS dataset. The first column corresponds to the original LVIS images and the rest of the columns are generations with different seeds. These generated images are then fed to our filtering pipeline for further processing before use in training.

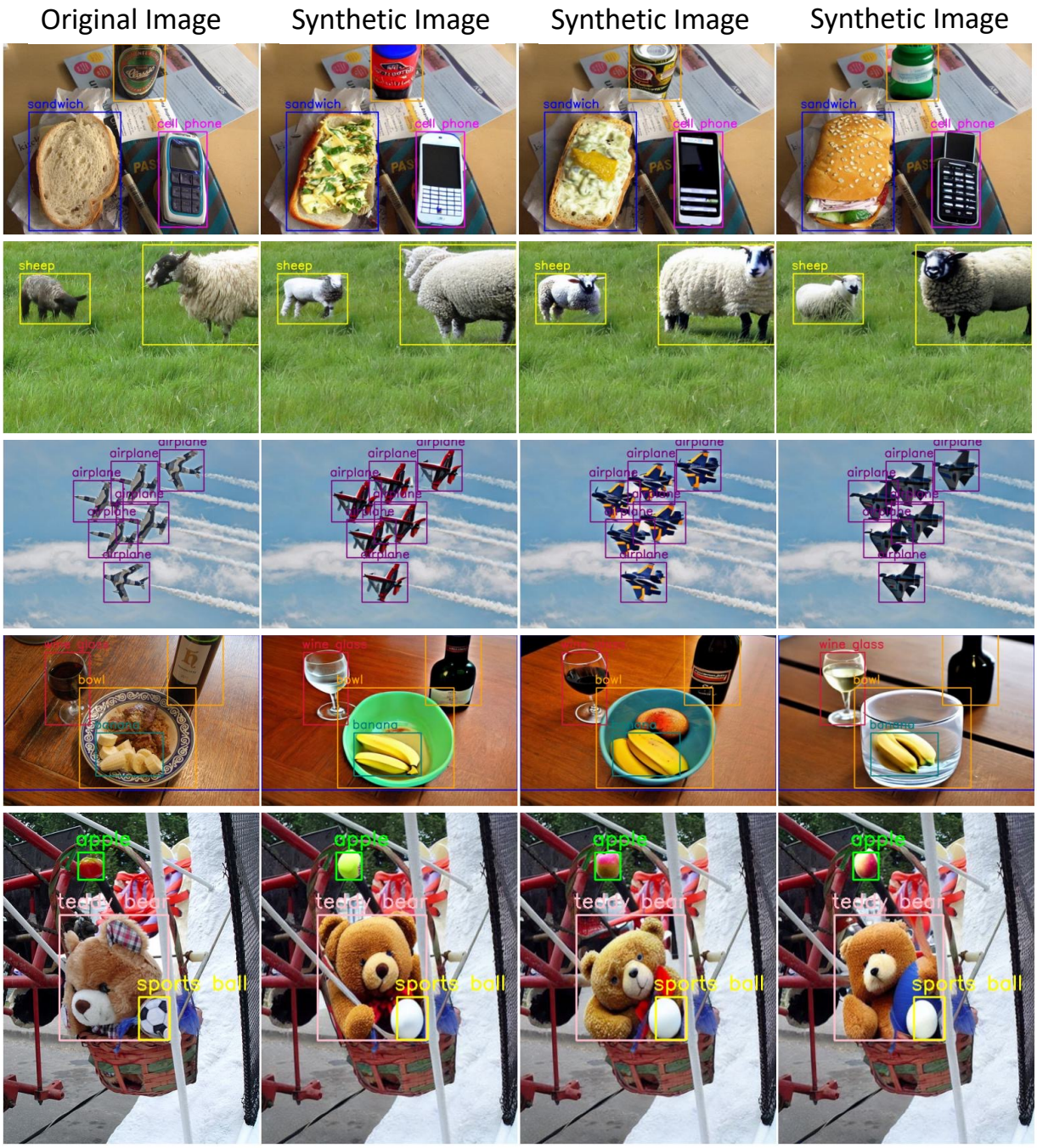


Figure 7.8: Examples of generations using the inpainting diffusion model on the COCO dataset. The first column corresponds to the original COCO images and the rest of the columns are generations with different seeds. These generated images are then fed to our filtering pipeline for further processing before use in training.

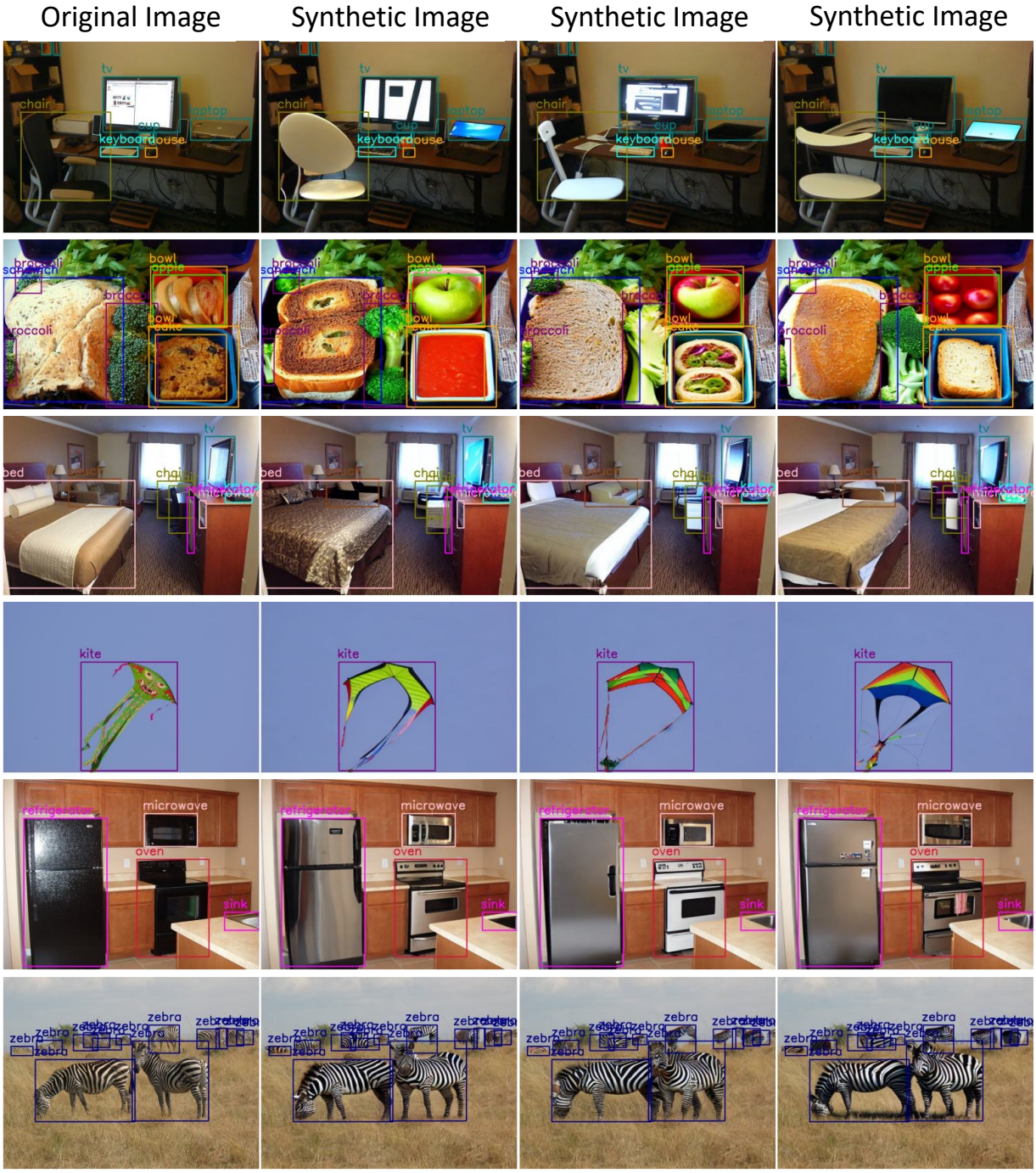


Figure 7.9: Examples of generations using the inpainting diffusion model on the COCO dataset. The first column corresponds to the original COCO images and the rest of the columns are generations with different seeds. These generated images are then fed to our filtering pipeline for further processing before use in training.

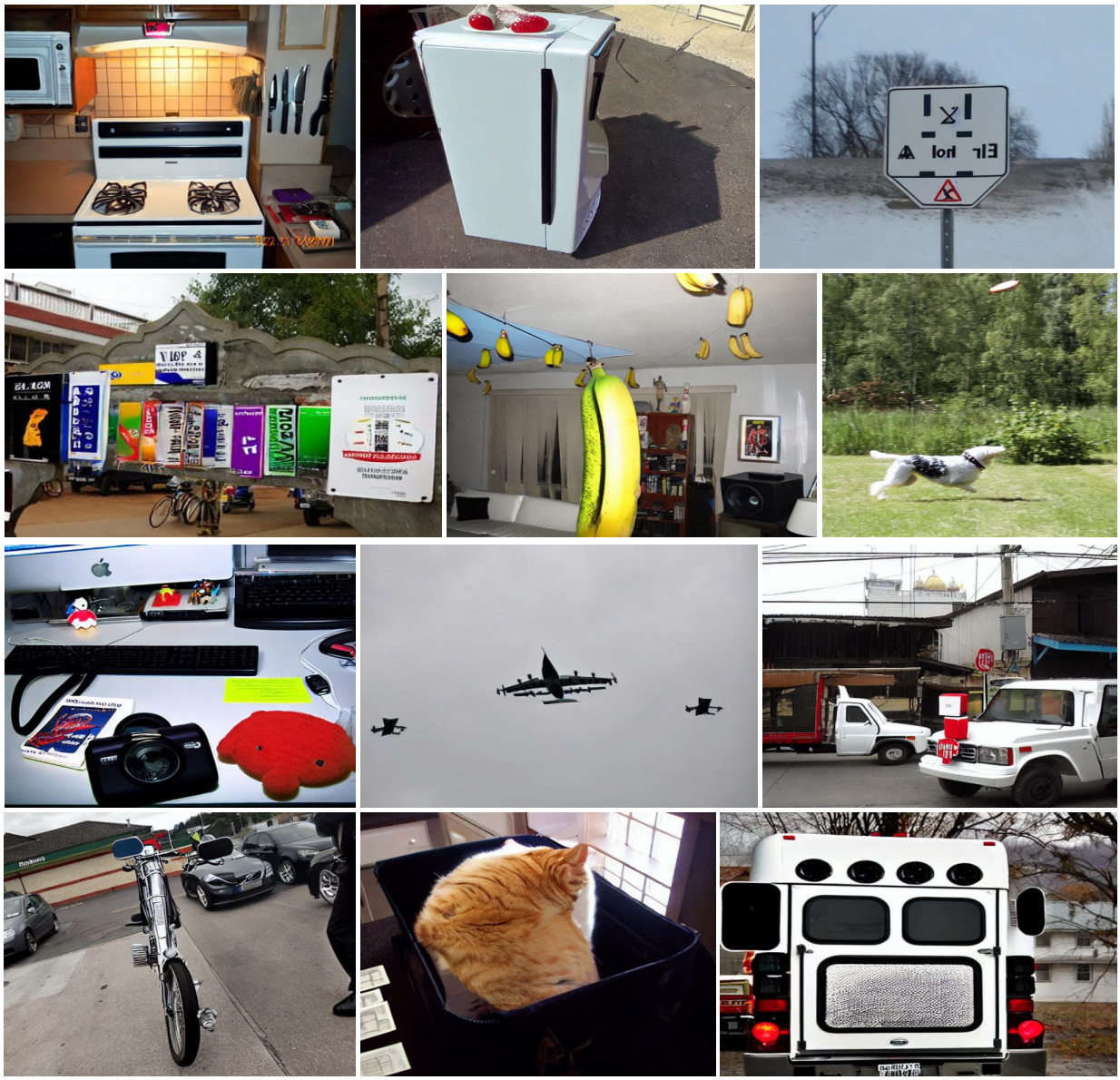


Figure 7.10: Examples of samples discarded during the image level filtering on LVIS. As can be seen, image level filtering is able to remove images with artifacts present at a global level.

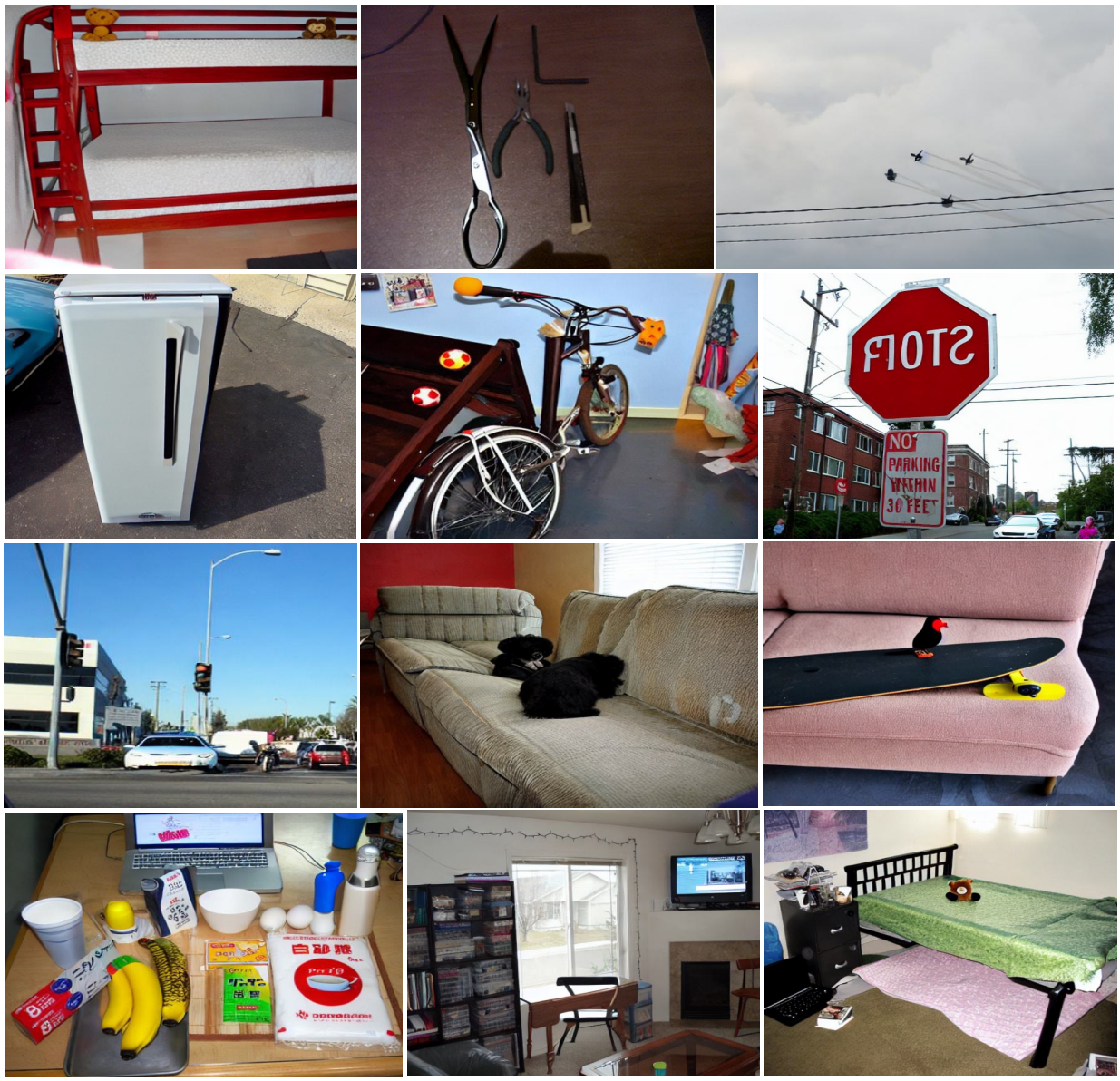


Figure 7.11: Examples of samples discarded during the image level filtering on COCO. As can be seen, image level filtering is able to remove images with artifacts present at a global level.

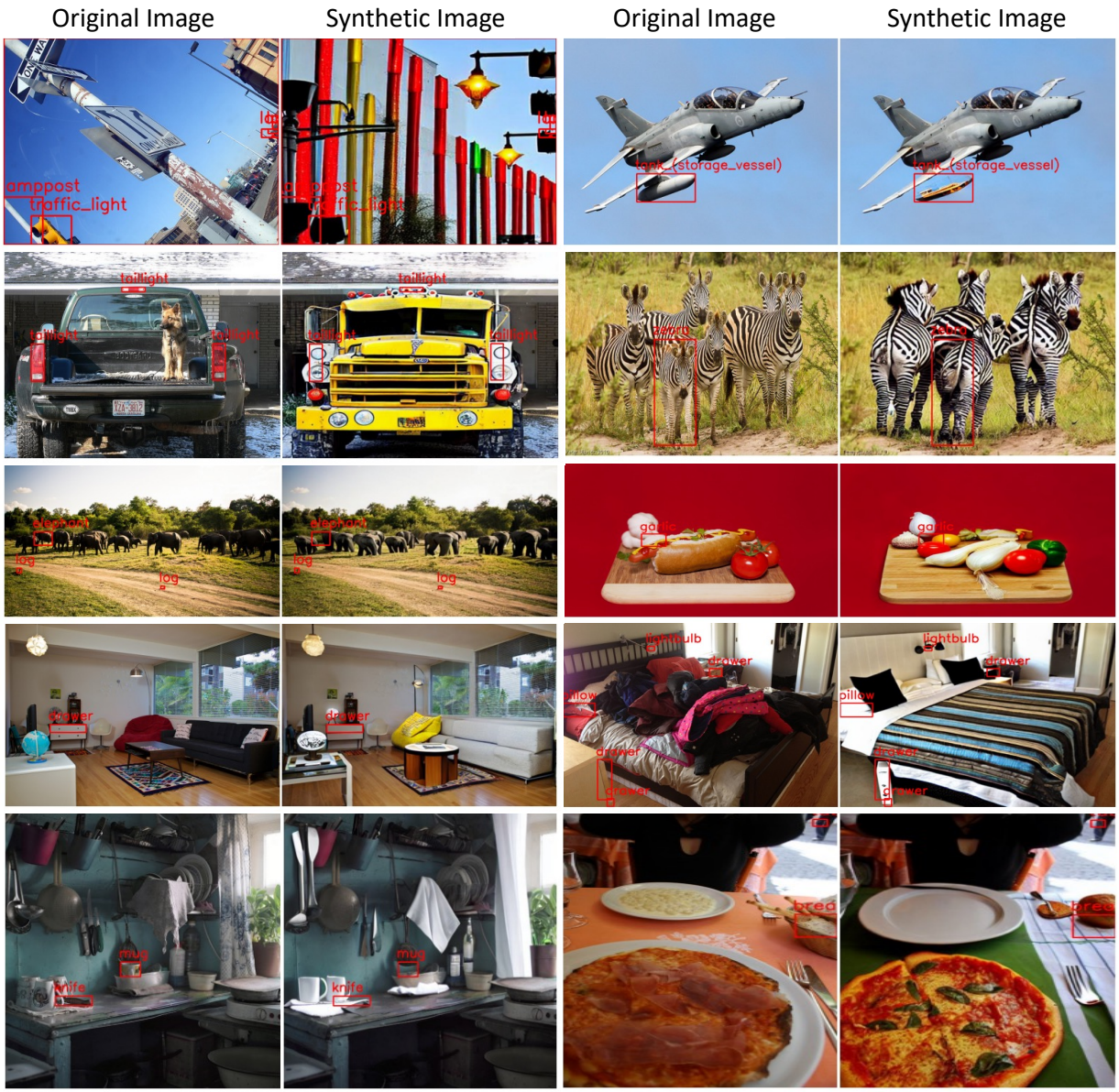


Figure 7.12: Examples of ground-truth instance annotations discarded by the detector filtering highlighted in red for LVIS dataset.

Figure 7.13: Examples of ground-truth instance annotations discarded by the detector filtering highlighted in red for COCO dataset.

Table 7.12: Comparisons across different τ_i on the LVIS dataset. We report the overall AP and also AP for rare (AP_r), common (AP_c) and frequent (AP_f) classes for both box and mask evaluations.

τ_i	Box				Mask			
	AP	AP_r	AP_c	AP_f	AP	AP_r	AP_c	AP_f
0.0	**24.42**	**15.43**	**22.63**	**30.38**	**23.67**	**15.33**	**22.62**	**28.51**
0.1	23.89	14.06	21.88	30.46	23.21	14.27	21.92	28.57
0.2	23.92	13.82	22.08	30.41	23.34	14.63	22.14	28.50
0.3	23.70	13.27	21.86	30.34	23.02	13.32	21.95	28.49
0.4	24.06	14.83	22.01	30.39	23.38	15.15	22.07	28.45
0.5	23.69	12.31	22.09	30.46	22.99	12.61	22.06	28.60

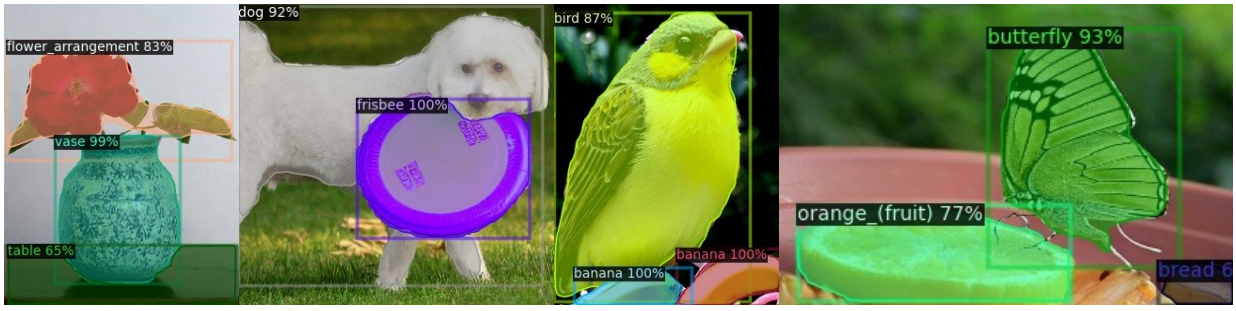


Figure 7.14: Predicted masks on synthetic data.

Figure 7.15: Images with different image level filtering score.

Chapter 8: LiFT: A Surprisingly Simple Lightweight Feature Transform for Dense ViT Descriptors

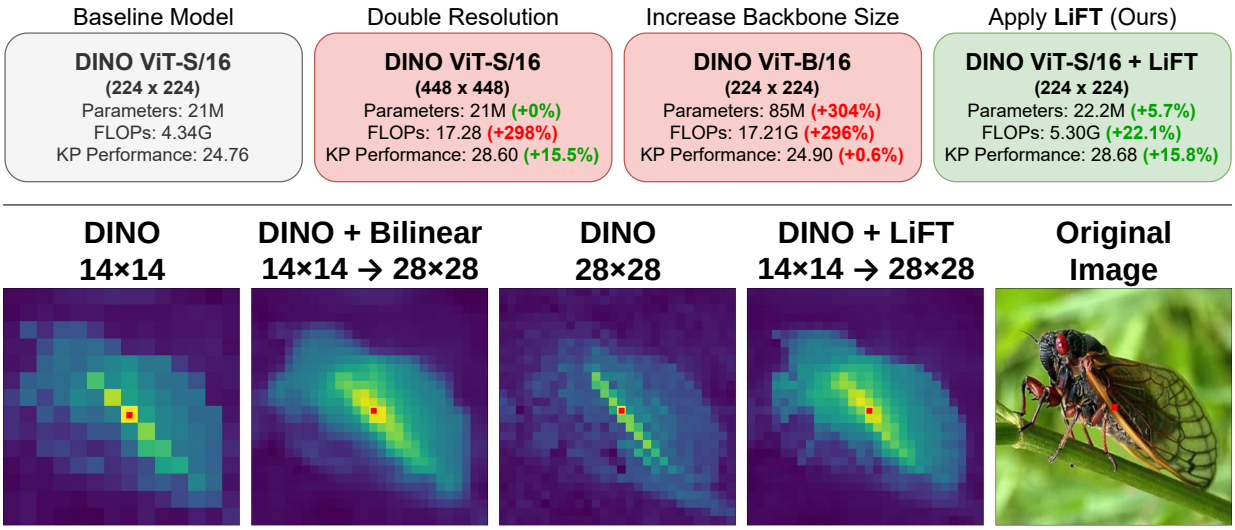


Figure 8.1: **(Top)** Increasing the backbone size or doubling the input resolution can boost the effectiveness of self-supervised ViT features for dense tasks like keypoint (KP) correspondence. However, both of these options come at a significant cost in terms of parameter count, inference cost, or both. We present **LiFT**, a surprisingly simple **Li**ghtweight **F**eature **T**ransform that unlocks the benefits of dense self-supervised ViT representations for minimal extra cost. **(Bottom)** LiFT also has useful emergent properties, such as yielding cleaner object boundaries in feature similarity maps.

In recent years, Vision Transformers (ViTs) [334] have emerged as preferred architectures for many image and video recognition tasks in the Computer Vision community. They also represent a major design shift compared with the well-explored Convolutional Neural Networks (CNNs). ViTs typically convert images into a very coarse grid of image patches (or tokens) before applying transformer layers. This allows ViTs to learn increasingly powerful patch-wise

representations in successive layers [335]. The expressive power of ViTs stems from their wide receptive field throughout all layers made possible by multi-headed self-attention operations [58]. The downside of this design is that despite being able to learn powerful representations, ViTs often lack spatial granularity in their features due to the low resolution of the token/patch grid. This hinders their off-the-shelf application to dense and local tasks such as object detection, segmentation, and keypoint correspondence. Increasing the feature resolution of a ViT directly using a larger image size or smaller patch size leads to an increased number of patches. Self-attention, being a quadratic operation, grows in memory consumption as $\mathcal{O}(N^2)$ where N is the number of patches in the image. Prior works have proposed alterations to the ViT architecture to make it better suited for dense tasks, but their methods either involve expensive carefully designed training, task-specific loss functions or heuristics, or high inference costs [55, 336, 337].

In this work, we propose a simple **Li**ghtweight **F**eature **T**ransform or **LiFT** to generate dense ViT features that provide significant performance gains in downstream tasks such as detection, segmentation, keypoint correspondence, and object discovery. LiFT can unlock the benefits of dense feature representations for a fraction of the computational cost compared with other approaches. As illustrated in Figure 8.2, our proposed method fuses the coarse high-level information of ViT features with convolution-based image features derived from the original image to generate higher-density feature maps without incurring the high computational cost of extra tokens. We show that this approach does not require any complex training recipe and, once trained on a general purpose dataset, generalizes well to multiple downstream tasks. Our approach can be trained with a simple self-supervised loss and generalizes to input image resolutions not seen during training. LiFT can be readily plugged on top of any ViT backbone to enhance its features, and it can also be integrated into pipelines that use additional task-specific downstream modules,

like the Mask-RCNN head [338] used by ViTDet [339]. Additionally, we show that we can apply LiFT in a recursive manner to increase feature resolution even further.

We demonstrate the effectiveness of LiFT quantitatively on 'local' tasks, which require features computed at precise locations, as well as on 'dense' tasks, which require features computed for the entire image. Specifically, we present results for LiFT applied to SPair-71k Keypoint Correspondence [72], COCO Detection and Segmentation [68], DAVIS Video Segmentation [71], and Unsupervised Object Discovery on Pascal VOC 2007 [100], Pascal VOC 2012 [1] and COCO20K [94]. For all of these tasks, LiFT is able to meet or exceed the performance of prior works for a fraction of the computational cost. As an example, in Figure 8.1 we compare three options for boosting performance in SPair-71k Keypoint Correspondence. LiFT provides a significant performance gain while increasing the total parameter count of the network by a mere 5.7%. This is compared to the 304% parameter count increase incurred by the step up from ViT-S/16 to ViT-B/16. Increasing the input resolution is a trivially easy way to boost the feature density, and it also gives improved performance. However, it increases the total inference FLOPs by almost 300% while LiFT only increases the cost by 22.1%, giving a far superior compute cost *vs*. performance trade off.

Despite the simplicity of our LiFT approach, we show that it is not just learning a more complex version of bilinear upsampling. Instead, we demonstrate that LiFT has several desirable emergent properties that enhance ViT features to make them better suited for dense tasks. We find that LiFT improves the scale invariance of ViT features, as measured using Centered Kernel Alignment (CKA) [77, 340]. We also qualitatively show that LiFT yields better object boundary maps when computing feature similarity maps. Overall, LiFT represents an orthogonal avenue of improvement compared to prior dense ViT feature extraction strategies, and it can be combined

with past methods to further advance self-supervised performance on dense prediction tasks. In summary, our contributions are as follows:

- We propose **LiFT**, a **Li**ghtweight **F**eature **T**ransform that boosts the performance of existing ViT features on dense and local downstream tasks using a simple, quick training and inference strategy.
- We show that LiFT boosts the performance of self-supervised ViT features for detection, segmentation, keypoint correspondence, and object discovery tasks.
- We demonstrate the adaptability of LiFT for any ViT backbone by showing improvements with DINO [26], MoCo [341] and Supervised ViT features. Additionally, LiFT even works on image resolutions not used during training.
- We show that LiFT features have desirable emergent properties like improved scale invariance and better feature alignment with object boundaries.

8.1 Related Work

8.1.1 Vision Transformers

Vision Transformers (ViTs) [334] have gained wide popularity as general-purpose models for multiple computer vision tasks such as image classification [334, 342, 343], object detection [59, 339, 344], segmentation [345, 346], video classification [347, 348], and more. Many of these methods adapt ViTs to different tasks by making suitable changes to the output heads [339, 349]. There have also been multiple variants of ViT like Swin [59], MViT [350], and PVT [351] which incorporate hierarchy and multiscale learning. Additionally, there are some

works which try to bridge the gap between transformers and CNNs by incorporating convolutions into ViT architectures [352, 353]. In this work, we focus on improving traditional or "Plain" ViT backbones, as they are the most general and widely adopted form of ViT. These models learn powerful representations, but they suffer in terms of feature resolution, an issue which we aim to address with LiFT.

8.1.2 Supervision Strategies for ViTs

Many works have proposed self-supervised tasks to learn meaningful representations with ViTs. These approaches do not require labels and can make use of large amounts of unlabeled data. These self-supervised models have powerful off-the-shelf features and can act as good pre-trained initializations for finetuning on downstream tasks. Among the most popular are approaches like Momentum Contrast methods (MoCo) [27, 33, 64] which enforce consistency between features for different augmentations of the same image. DINO [26] also utilizes a similar self-supervised approach, but uses different crops along with other augmentations. MAE [28] learns to predict masked-out image regions, and there are works like CLIP [62] which match text and image embeddings to learn semantically rich features without direct label-level supervision. Prior works [57, 335, 354] have shown that these differences in pre-training lead to significant differences in the properties of the learned features. We show that LiFT can improve the quality and usefulness of ViT features for a range of different pretraining methods.

8.1.3 Feature Densification

Works like [55] and [335] show the general benefits of dense feature maps for local tasks, and multiple works have been proposed to extract denser feature arrays from pretrained networks. [355] proposes a GAN-based approach for CNN feature densification which requires careful training and a mixture of adversarial and focal loss. In comparison, our LiFT approach is easy to train through a simple self-supervised objective, and it does not require a discriminator module or careful loss tuning. ViT-Adapter [356] applies ViTs to dense tasks through the use of finetunable side-networks for feature pyramid extraction. Like ViT-Adapter, LiFT aims to enhance "Plain" ViT features for dense tasks, however, the ViT-Adapter method is not task-agnostic, and it is trained in a fully supervised way with full detection and segmentation labels for the downstream dataset. In contrast, LiFT is a task-agnostic general enhancement for ViT features, and it is trained with a completely self-supervised objective. LiFT is also faster and easier to train than ViT-Adapter, as it does not require passing gradients through the ViT backbone, and it is also $\sim 4.8\times$ smaller than the similar ViT-Adapter.

Other works, like [357] and [358], have applied student/teacher distillation with feature super-resolution to improve CNN classification performance on low-resolution images. While we also perform feature super-resolution, the aim of our work is not to distill a student network, but rather we aim to generate densified features while completely avoiding finetuning the ViT backbone. [55] proposes a simpler technique to increase the density of ViT features by reducing the stride during initial image patch extraction. This does not require any training, but also becomes computationally expensive as the number of tokens increase quadratically requiring more GPU memory and FLOPs. Our LiFT approach can be thought of as a "shortcut" to achieve

the benefits of denser features for a fraction of the computation cost of increased tokens.

8.1.4 Finetuning ViTs for Dense Tasks

The prior works most similar to LiFT are those that apply self-supervised finetuning to pretrained ViTs to improve their features for dense tasks. SelfPatch [337] and Leopart [336] both use pretrained DINO models as a starting point and improve their patch-level representations using dense self-supervised tasks. SelfPatch learns by enforcing similarity to neighboring patch features, and Leopart uses spatially dense clustering to enforce similarity within clusters to learn better part representations. These approaches finetune the full backbone with specialized training strategies and losses, which can still be expensive to train and not easily extendable to other backbones. In contrast, our approach does not require backbone finetuning at all, and instead trains a lightweight post-processing module. LiFT can be easily trained on ImageNet [66] in a self-supervised manner in very few epochs and afterwards it can generalize to multiple downstream tasks.

8.2 Method

8.2.1 ViT Background

Consider a ViT that takes an image with dimensions $H \times W \times 3$ as input and outputs feature descriptors at the resolution $\frac{H}{P} \times \frac{W}{P}$, where P is the patch size. P is usually 8 or 16 depending on the ViT variant. We typically assume that the patch-extraction stride length, S, is equal to P, though this can be altered, as proposed by [55]. For now we will assume that $S = P$. Our goal is to transform the coarse, low-resolution features of a pretrained ViT into

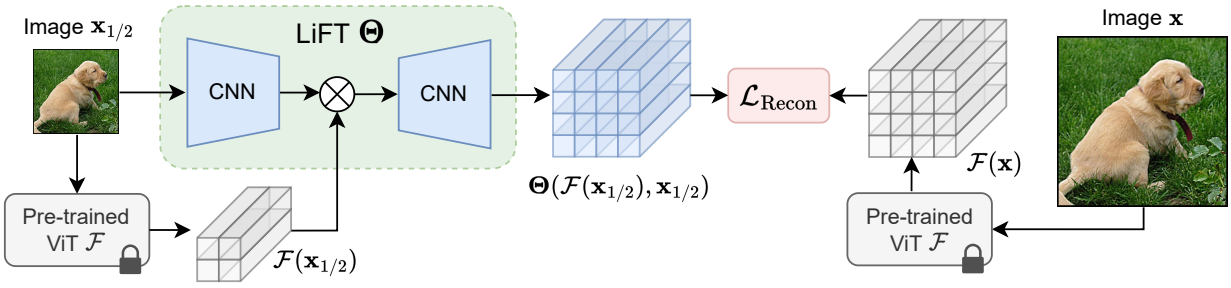


Figure 8.2: Illustration of **LiFT**, our proposed **Li**ghtweight **F**eature **T**ransform for generating dense ViT descriptors. The frozen ViT backbone is used to extract features for both low- and high-resolution images. The low-resolution image and its corresponding features are passed through LiFT, which generates a dense version of the features. The LiFT Block first encodes fine-resolution image features using a small CNN. It then combines the CNN features with the ViT features at multiple phases in an upsampling CNN, which outputs dense features. The LiFT block is trained using a self-supervised reconstruction error with the corresponding high-resolution features.

dense feature descriptors without having to re-train or finetune the ViT. A naïve way to achieve this transformation could be to scale up the input image to $CH \times CW$ resolution to achieve a feature grid that is C times larger along both dimensions. This approach can result in a significant increase in memory consumption and can be prohibitively expensive since the memory of the ViT scales with $\mathcal{O}(H^2W^2)$. Another option could be to upscale the features directly by a factor of C using bilinear-interpolation. This approach computes sub-optimal pixel-level features as it simply bilinearly redistributes the features between the centers of patches. Such an approach fails to take advantage of the information that is readily available in the original image space. LiFT takes advantage of this information.

8.2.2 Lightweight Feature Transform (LiFT)

Our proposed approach, LiFT, builds on the hypothesis that, even though the ViT feature descriptors have a low spatial resolution, their high dimensionality allows them to store rich information about the image structure. This hypothesis is further supported by works on internal

learning of images [359, 360, 361]. However, unlike internal learning, we propose to train a general-purpose, lightweight upsampling network using only self-supervision to double the resolution of feature descriptors obtained from a frozen, pretrained ViT. Furthermore, we can gain additional fine-level information directly from the original image at the same resolution used to generate the ViT features. We fuse these two information sources to create our final LiFT module, as illustrated in Figure 8.2.

Given ViT descriptors of size $\frac{H}{P} \times \frac{W}{P}$, a single LiFT expansion block scales them to $\frac{2H}{P} \times \frac{2W}{P}$ in a single forward pass. Our LiFT block is built following a U-Net-style structure [235] with skip connections, where semantically rich but coarse ViT features are combined with shallow but dense image features derived from a second input with the original image. The image input is processed through a series of convolution blocks and the resulting features are concatenated to the ViT features. Then, a single transpose convolution block is applied to generate the upscaled semantically rich features. Thanks to its fully convolutional nature, the LiFT block can be applied to any image size. One can even apply a LiFT block multiple times to further upscale the features, as shown in Section 8.5.4.

8.2.3 Training Objective

Given an image $\mathbf{x} \in \mathbb{R}^{H \times W \times 3}$ and a pretrained, frozen ViT model $\mathcal{F}$ of stride P, we extract its features from the last layer such that $\mathcal{F}(\mathbf{x}) \in \mathbb{R}^{\frac{H}{P} \times \frac{W}{P} \times D}$, where D is the feature dimension. The LiFT block $\mathbf{\Theta}$ upscales the features from resolution $\frac{H}{P} \times \frac{W}{P}$ to $\frac{2H}{P} \times \frac{2W}{P}$. For training, we

propose the following multi-scale reconstruction objective:

$$\mathcal{L}_{\text{Recon}} = \quad d\left(\mathcal{F}(\mathbf{x}), \mathbf{\Theta}(\mathcal{F}(\mathbf{x}_{1/2}), \mathbf{x}_{1/2}\right) + d\left(\mathcal{F}(\mathbf{x}_{1/2}), \mathbf{\Theta}(\mathcal{F}(\mathbf{x}_{1/4}), \mathbf{x}_{1/4}\right) \tag{8.1}$$

Where $\mathbf{x}_{1/2}$ and $\mathbf{x}_{1/4}$ are the images resized to $1/2$ and $1/4$ of the original image resolutions, and d is a distance function. For d we select the cosine distance metric, as it inherently normalizes the output and empirically achieves better performance. When processing $\mathbf{x}_{1/2}$ and $\mathbf{x}_{1/4}$, we follow the method of [26] to handle positional embeddings for images of different sizes. We optimize the parameters of the LiFT module $\mathbf{\Theta}$ to minimize Equation 8.1.

8.2.4 Training Details

Training LiFT is fast and efficient, as it is lightweight and does not require propagating gradients through the ViT backbone. The LiFT module, $\mathbf{\Theta}$, is a small network with $1.1M$ trainable parameters. We train LiFT for 5 epochs on the ImageNet dataset on a single GPU. We use a learning rate of 0.001 with a batch size of 256 for stride 16 training. We use DINO [26] ViT-S/16 as our base ViT, and we apply color jitter as the only augmentation. In total, training takes only $\sim$8 hours on one RTX A6000 GPU. Once our LiFT module is trained, it is a general purpose feature enhancement module that can be directly applied to a range of downstream tasks without need for any further finetuning of LiFT or the frozen ViT backbone. We demonstrate the flexibility of LiFT by applying features from the same DINO+LiFT model to several tasks in Sections 8.3.1, 8.3.2, and 8.3.3.

8.2.5 Using LiFT with Downstream Modules

Our LiFT module can be easily applied to downstream tasks that directly operate on the output image features. However, LiFT can also be applied in circumstances where additional downstream modules follow the feature extracting backbone. To demonstrate this, we show how to apply LiFT to the ViTDet architecture [339], which combines a ViT backbone with a Mask-RCNN-style head [338], to perform COCO Object Detection and Segmentation. To achieve this, we inject the LiFT module after the backbone and before the Mask-RCNN head in a pretrained ViTDet model. We train this LiFT module on COCO training data using the same self-supervised objective described in Section 8.2.3. One can then optionally finetune the downstream head on the LiFT-generated features. We show that finetuning the head is preferable, but even without finetuning, adding LiFT is beneficial. We present these results in Section 8.3.4.

8.2.6 Baseline Methods

We propose LiFT as a self-supervised, task-agnostic, general enhancement for ViT features. For baselines, we compare with the most similar prior works, SelfPatch [336] and Leopart [337], both of which finetune the entire DINO model on custom self-supervised losses to improve the quality of the spatial features for dense tasks. We demonstrate the effectiveness of LiFT compared with these methods for a range of dense and local tasks in Section 8.3. In addition, we compare with the reduced token stride strategy of [55] and show the improved computational efficiency of LiFT in Section 8.4. Note that LiFT represents an orthogonal direction of ViT improvement that can be used in combination with methods like [55] and [337], which we show in the following sections.

Table 8.1: Comparison between LiFT and other baselines on the keypoint correspondence task on SPair-71k. We report PCK@0.1 and 0.05 at multiple input resolutions.

	PCK@0.1				PCK@0.05			
Method/Resolution	56	112	224	448	56	112	224	448
DINO	2.04	12.67	24.76	28.6	0.51	3.61	9.54	15.33
Leopart	2.35	11.2	23.33	26.54	0.6	3.22	8.9	12.26
SelfPatch	2.13	12.18	23.03	27.34	0.44	3.61	9.32	14.44
DINO+LiFT	**5.05**	**17.72**	**28.68**	**31.38**	**1.19**	**6.29**	**14.72**	**18.90**

8.3 Performance Benefits of LiFT

8.3.1 SPair Keypoint Correspondence

This task involves matching keypoints between pairs of images in the SPair-71k dataset. We follow the evaluation protocol of Amir et al. [55] and report Percentage of Correct Keypoints (PCK) as the metric. We extract dense features using the frozen DINO+LiFT combination trained in Section 8.2.4. For brevity, we report results at PCK thresholds of 0.1 and 0.05 in the main paper but we also see consistent improvements at the 0.01 threshold For this task, the features of all methods are bilinearly interpolated to match the original image resolution before feature matching begins. Table 8.1 presents the results. Compared to both the base DINO model and finetuned approaches, LiFT performs the best across all resolutions for both PCK@0.1 (left) and PCK@0.05 (right). For lower resolutions like 56×56, LiFT more than doubles the performance on both metrics. Also note that under the PCK@0.1 metric, DINO+LiFT at 224×224 resolution beats all non-LiFT approaches run at the higher resolution of 448×448, even though both configurations produce final features of the same density.

Table 8.2: Comparison on the DAVIS video object segmentation task. We report results for the J and F mean.

Method/Resolution	56	112	224	448
DINO	7.4	17.5	33.0	50.9
Leopart	6.9	16.1	30.3	45.1
SelfPatch	7.4	17.2	33.0	51.4
DINO+Bilinear	10.8	23.7	37.0	53.0
DINO+LiFT	**13.0**	**28.0**	**44.3**	**61.1**

Table 8.3: When added to ViTDet, LiFT boosts detection and segmentation performance on COCO. ViTDet+LiFT* shows results without fine-tuning the Mask-RCNN head.

	Detection			Segmentation		
Method	AP	AP_{50}	AP_{75}	AP	AP_{50}	AP_{75}
ViTDet	39.50	61.56	42.23	37.81	60.47	39.97
ViTDet+LiFT*	42.77	61.05	46.60	38.18	58.28	40.96
ViTDet+LiFT	**45.98**	**66.41**	**49.87**	**40.78**	**63.34**	**43.57**

8.3.2 DAVIS Video Object Segmentation

This task involves propagating a video object segmentation across multiple frames where the first frame ground truth segmentation mask is provided. This is achieved through dense feature matching between frames. Again, we extract dense features with the same pre-trained DINO+LiFT. We follow the evaluation protocol of [79] and, for brevity, we report results for the J&F mean metric in the main paper, but we see consistent improvements across the J mean and F mean individually. In Table 8.2, it can be seen that across resolutions and comparison methods, LiFT outperforms all other approaches. At the lowest resolution of 56×56, the performance gain over the base DINO is $\sim 2\times$. On average, we improve by 9.4 points over base DINO.

8.3.3 Unsupervised Object Discovery

We test the benefits of LiFT for Unsupervised Single Object Discovery on PASCAL VOC 2007 [100], PASCAL VOC 2012 [1], and COCO20K [94]. We apply TokenCut [362], which performs Graph Cut on features, to LiFT and the other baseline methods. Similar to prior works [362, 363, 364, 365, 366, 367] we report the Correct Localization (CorLoc) metric, which is computed as the fraction of the images in which at least one object box prediction has an IoU

Table 8.4: Unsupervised Object Discovery comparison on PASCAL VOC 2007, PASCAL VOC 2012, and COCO20K. We report results for the CorLoc metric.

Dataset	Method	Resolution			
		56	112	224	448
VOC07	DINO	20.74	50.07	65.60	68.27
	Leopart	18.92	32.59	51.59	48.79
	SelfPatch	18.04	41.99	62.40	63.62
	DINO+LiFT	**36.54**	**62.02**	**68.79**	**69.65**
VOC12	DINO	23.27	55.33	69.01	71.64
	Leopart	22.44	37.41	55.74	54.40
	SelfPatch	20.19	47.32	68.02	66.48
	DINO+LiFT	**40.56**	**66.21**	**70.91**	**71.71**
COCO20K	DINO	16.28	40.08	53.98	57.99
	Leopart	16.14	26.78	43.89	44.08
	SelfPatch	14.15	35.76	52.18	55.47
	DINO+LiFT	**27.72**	**50.20**	**58.03**	**60.50**

greater than a threshold (0.5) with a ground truth box. As shown in Table 8.4, LiFT gives a good boost in CorLoc, with gains across all resolutions and with LiFT outperforming the other approaches. It should be noted that the performance gains at lower resolutions are especially large. For example, at the 56×56 resolution, we see an improvement of 15.8, 17.29, and 11.44 on CorLoc for VOC07, VOC12, and COCO20K respectively.

8.3.4 COCO Detection and Segmentation

To further demonstrate the versatility of LiFT, we show how it can be applied to COCO Detection and Segmentation using a ViTDet+LiFT model as described in Section 8.2.5. As LiFT is trained to emulate the same feature distribution generated for higher resolution inputs, the LiFT-generated features can be directly input into the downstream Mask-RCNN head without any additional training. However, we find that performing limited fine-tuning of the head can be

beneficial. In Table 8.3, the combination of ViTDet+LiFT gives a $\sim 6.5\%$ boost for Detection AP and a $\sim 3\%$ boost for Segmentation AP when the downstream head is finetuned. Even without head finetuning, denoted as ViTDet+LiFT*, we see a small performance gain for AP and AP_{75} in both Detection and Segmentation.

8.4 Computational Efficiency of LiFT

We have shown that feature densification with LiFT provides significant performance benefits in several tasks. We now show why LiFT is a *Lightweight* transform, as it is vastly more computationally efficient than other alternatives for feature densification. A trivially easy way to boost the density of ViT features is to increase the resolution of the input image, which increases token density but also computational cost. Another option is the dense feature extraction strategy of [55] which increases the number of tokens in the network by reducing the stride during patch extraction. We present a comprehensive compute cost *vs*. performance benefit analysis for LiFT and alternative methods, and we show that LiFT acts as a "shortcut" to achieve higher resolution features for minimal extra compute cost. We also show that LiFT can be combined with these methods for further improved performance in exchange for higher compute.

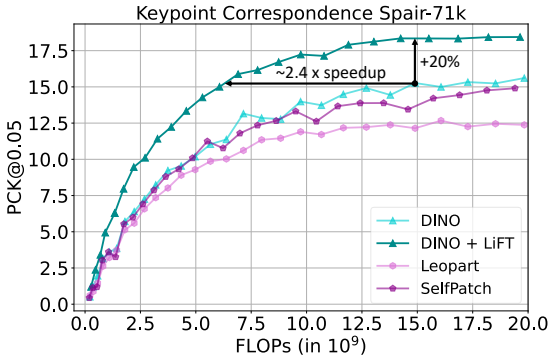
8.4.1 LiFT, Resolution, and Stride

For this analysis, we take SPair Keypoint Correspondence and DAVIS Video Segmentation as the tasks, and we measure compute cost in FLOPs (G) against performance both with and without LiFT over a range of input resolutions and strides. As shown in rows 1 and 2 of Table 8.5, LiFT introduces a small increase in FLOPs from 4.34G to 5.30G ($+22\%$) but also brings a sig-

Table 8.5: FLOPs comparison for a single forward pass across resolutions and strides for base DINO and LiFT. Along with FLOPS, we report the performance on both SPair-71k and DAVIS.

Method	Res.	Stride	FLOPs (G)	PCK@0.1	J&F mean
DINO	224	16	4.34	24.76	33.0
DINO+LiFT			5.30	28.68	44.3
DINO		8	16.07	29.92	43.9
DINO+LiFT			19.65	31.91	52.6
DINO	448	16	17.28	28.60	50.9
DINO+LiFT			21.12	31.38	61.1
DINO		8	66.60	31.92	61.9
DINO+LiFT			81.18	32.20	69.7

Figure 8.3: Performance *vs.* Compute Cost trade-off curve for SPair-71k keypoint correspondence. For any given FLOP-budget, DINO+LiFT achieves far superior performance.



nificant performance improvement of 3.92, 5.18, and 11.3 points on the PCK@0.1, PCK@0.05, and J and F Mean metrics respectively. For comparison, reducing the stride from 16 to 8 (row 1 *vs.* row 3) gives a similar level of improvement, but nearly quadruples the FLOPs from 4.34G to 16.07G ($+270\%$). A similar trend of large improvements can be seen when comparing pairs of rows (3 *vs.* 4, 5 *vs.* 6) with DINO *vs.* DINO+LiFT. The best overall results are achieved by combining LiFT with increased resolution and reduced stride, but this also comes with the highest computation cost. It should also be noted that DINO+LiFT at 224×224 resolution and stride 8 (row 4) uses $3\times$ less FLOPs than DINO at 448×448 resolution at stride 8 (row 7) while having similar performance on the PCK metrics.

8.4.2 Cost *vs.* Performance Trade-off Curve

Next, we present a comprehensive Compute Cost *vs.* Performance trade-off analysis on SPair Keypoint Correspondence. By incrementally increasing the input resolution, we can gradually increase both the performance and inference cost for DINO+LiFT and the baseline methods

Table 8.6: Comparison of parameters & FLOPs *vs.* performance at 224×224 resolution for Keypoint Correspondence. We report PCK@0.1 and PCK@0.05 on SPair-71k.

Method	Parameters	FLOPs (G)	PCK@0.1	PCK@0.05
DINO S/16	21M	4.34	24.76	9.54
DINO S/16+LiFT	22.2M	5.30	**28.68**	**14.72**
DINO B/16	85M	17.21	24.90	9.64

Table 8.7: LiFT when using different backbones for training, inference, or both. We report PCK@0.1 on SPair-71k.

		Training Model		
Inference Model	No LiFT	DINO	MoCo	ViT
DINO	28.6	**31.38**	16.02	20.71
MoCo	12.31	9.86	**16.34**	11.31
ViT	16.9	12.55	8.91	**18.69**

DINO, Leopart, and Selfpatch. As shown in Figure 8.3, we find that LiFT significantly outperforms all other methods at any given FLOP allowance, in most cases seeing a $\sim 20\%$ performance gain. Alternatively, LiFT can be run at a lower input resolution to achieve equivalent performance at a fraction of the compute cost. For example, to achieve a score of 15.0 in PCK@0.05, DINO+LiFT only requires ~ 6.25 Giga-FLOPs of compute power, while the other baselines require at least 15 Giga-FLOPs. At any point on the trade-off curve, DINO+LiFT far surpasses the alternatives.

8.4.3 Parameter Count

We acknowlege that the addition of the LiFT module slightly increases the overall model size and parameter count as shown in Table 8.6. However, this addition is quite small and only represents a $+5.7\%$ change in total parameters. For comparison, the jump from ViT-S to ViT-B results in a $+304\%$ increase in parameters. Furthermore, for dense tasks like SPair Keypoint Correspondence, we find that the performance benefits provided by LiFT far exceed the benefits of a larger backbone. Note that the methods Leopart and SelfPatch do not increase the parameter count of the ViT, as they finetune the DINO backbone instead of introducing new modules. However, we believe the major performance benefits of LiFT strongly justify the small extra costs

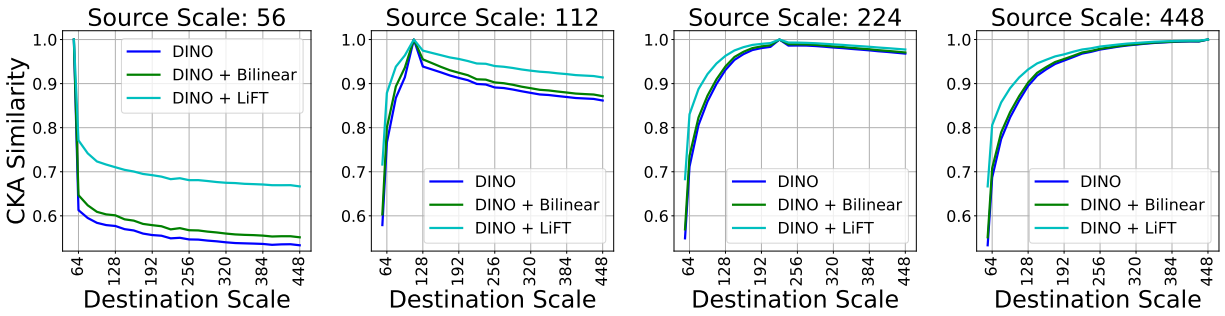


Figure 8.4: CKA Similarity of ViT features extracted from SPair-71k images at different input image sizes, denoted by Source Scale and Destination Scale. LiFT produces features that are more scale-invariant, especially for smaller scale inputs and objects.

in Parameter Count and Inference FLOPs for a given resolution. And as shown in the previous section, for any fixed FLOP-budget, DINO+LiFT achieves far superior performance.

8.5 Properties of LiFT

8.5.1 LiFT and Scale Invariance of Features

In this experiment, we demonstrate that LiFT intrinsically learns to generate features that are more scale-invariant. We use the Centered Kernel Alignment (CKA) metric [76, 77, 87], which can measure the similarity of a pair of feature maps even when they are different sizes. Using images in the SPair-71k training set, we re-scale each image to a range of different sizes and then extract features with DINO or DINO+LiFT and measure the CKA similarity between all input size pairings. As a baseline, we also compare with bilinearly upsampled DINO features. In Figure 8.4, we take four source scales and plot the CKA similarity with the features at all other scales. We see that LiFT greatly improves the inter-scale feature similarity for small input scales, and moderately improves the similarity for medium and larger scales. This shows that LiFT produces representations that are more scale-invariant than the base DINO features. Bilinear

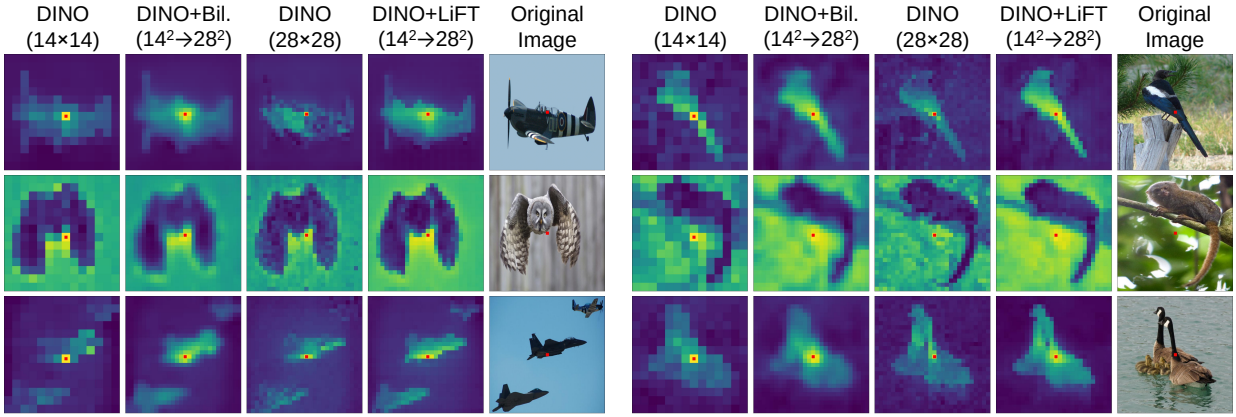


Figure 8.5: Visualization of the self-similarity of features for DINO, DINO + Bilinear interpolation, DINO with higher resolution image, and DINO + LiFT. To generate this visualization, the self-similarity is computed using the feature corresponding to the center of the grid (marked in red) and all other features from each spatial location. Brighter map shows a higher similarity. Best viewed digitally in color.

upsampling does provide a small improvement in CKA similarity across scales, though the benefit is far smaller than that of LiFT. This scale invariance property of LiFT is learned automatically, and likely comes from its multi-scale reconstruction objective. LiFT must learn to counteract the effect of input scale in order to map features from low resolution inputs to those of high resolution inputs. This property is desirable for dense tasks where objects appear at different scales.

8.5.2 Enhanced Self-Similarity Maps with LiFT

To further improve our understanding of the dense features generated by LiFT, we visualize the self-similarity of the features in Figure 8.5. Base DINO S/16 yields 14×14 features for a 224×224 image. In our comparisons, we visualize these features alongside: a bilinearly interpolated upsampling of these same features to 28×28 (DINO+Bilinear), DINO features for a 448×448 input image, and finally DINO+LiFT features generated for a 224×224 input image. The last three configurations all yield 28×28 feature grids. To visualize these features, we select the center-most token feature for each of the maps and compute its similarity with all other

features and visualize the similarity scores.

We show six feature similarity maps in Figure 8.5. We find that, qualitatively, the LiFT output gives a cleaner boundary and better highlights the content corresponding to the central patch. Having a high similarity to relevant regions and a clear boundary is beneficial for multiple localized downstream tasks. It also indicates that the DINO+LiFT features have better spatial awareness of object boundaries. Note that for row 2, where the central pixel corresponds to the background region, the similarity map highlights the background in the image, though it still appears that DINO+LiFT produces better similarity maps with sharper edges. In row 3, when there are multiple objects of the same type, DINO+LiFT better highlights the separate object instances. These results are qualitative, but they suggest that the LiFT-enhanced features have better content and boundary information than the base DINO features, which likely contributes to their improved performance in correspondence and segmentation tasks.

8.5.3 Variations in Backbone

One of the requirements of a feature densifying approach is that it should be easy to train on any backbone. To this end, we show that our approach consistently gives a performance gain with multiple different backbones: DINO (Table 8.8 row 1 *vs.* row 2), MoCo (Table 8.8 row 4 *vs.* row 5) and a fully-supervised ViT (Table 8.8 row 7 *vs.* row 8). This consistent improvement shows that LiFT can be trained in the exact same manner on differently trained ViTs without need for careful hyperparameter tuning. To verify that LiFT does not simply learn a bilinear interpolation, we apply LiFT modules trained on one backbone to the output of a different backbone. We show these results in Table 8.7 using the PCK@0.1 metric and 448×448 images. It can be seen

Table 8.8: Performance for LiFT with different backbones and for repeated application of LiFT. We report PCK@0.1 on SPair-71k.

Row	Backbone	Method	Resolution			
			56	112	224	448
1	DINO	-	2.04	12.67	24.76	28.6
2		LiFT	5.05	17.72	28.68	**31.38**
3		2×LiFT	**7.42**	**20.12**	**29.45**	31.35
4	MOCO	-	1.27	3.43	7.37	12.31
5		LiFT	6.48	10.51	14.13	16.34
6		2×LiFT	**8.72**	**13.21**	**16.12**	**17.08**
7	ViT	-	1.26	5.72	13.23	16.9
8		LiFT	3.76	9.21	**16.5**8	**18.69**
9		2×LiFT	**5.17**	**9.89**	16.49	18.18

that when LiFT is applied to a different model than what was used to train it, the performance drops and is lower than not applying LiFT. This shows that LiFT learns a model-specific feature-densifying transform and not a simple interpolation.

8.5.4 Repeated Application of the LiFT Module

In our base approach, the LiFT module is applied once to the ViT features to double their resolution. In this section, we also test the potential benefits of applying the LiFT block multiple times. To check this, we take the LiFT module and super-resolute the features twice by passing the output of first super-resolution again through the same LiFT network. We denote this approach as '2×LiFT' in Table 8.8. We can see that in most cases the performance increases after recursively applying LiFT. This is especially true for the 56×56, 112×112, and 224×224 resolutions. For 448×448, there are a few places where it negligibly drops performance, but it still shows improvement for most cases.

8.6 Discussion

We have presented **LiFT**, a simple yet effective self-supervised **Li**ghtweight **F**eature **T**ransform to boost the density of features of pretrained ViT backbones. This approach allows us to extract higher resolution spatial features from ViTs which can then be used for multiple dense downstream tasks. LiFT is task-agnostic and gives significant boosts in SPair Keypoint Correspondence, DAVIS Video Object Segmentation, Unsupervised Object Discovery, and COCO Detection and Segmentation. This benefit comes for a fraction of the computational cost compared with other densification methods. LiFT is a lightweight module that is easily trained with a self-supervised objective, and it is far cheaper than full backbone fine-tuning, as is done by other prior works. Through extensive experiments, we have shown that LiFT can be trained easily on any backbone and consistently leads to improved performance by generating better quality, higher-density features. We also show that LiFT can be applied to its own output in a recursive manner enabling good performance with even lower image resolutions. Finally, we show that our surprisingly simple method has several desirable emergent properties including scale-invariant features, and better object boundary maps. This makes LiFT a useful multipurpose tool for many potential downstream applications.

8.7 Appendix

8.7.1 Ablation Study of LiFT Design Choices

We present a careful analysis of LiFT design configurations by varying different factors in both training and inference. To show the general applicability and benefits of LiFT, we include

Table 8.9: Ablation of different design decisions for LiFT training for three different ViT backbones. We report PCK@0.1 and PCK@0.05 on SPair-71k. For each backbone, we mark the best score for each metric and input resolution in **bold**.

		PCK@0.1				PCK@0.05			
Row	Method/Resolution	56	112	224	448	56	112	224	448
1	DINO	2.04	12.67	24.76	28.6	0.51	3.61	9.54	15.33
2	DINO + Random LiFT	1.45	2.37	4.21	6.16	0.35	0.7	1.41	2.35
3	DINO + LiFT No Img.	4.38	15.74	28.49	**31.42**	1.14	5.03	13.28	18.33
4	DINO + LiFT L1	4.48	16.64	27.77	31.03	1.01	5.93	13.88	18.09
5	DINO + LiFT L2	4.82	**17.72**	28.17	31.13	**1.29**	6.18	14.12	18.37
6	DINO + LiFT	**5.05**	**17.72**	**28.68**	31.38	1.19	**6.29**	**14.72**	**18.90**
7	MOCO	1.27	3.43	7.37	12.31	0.21	0.84	2.35	5.49
8	MOCO + Random LiFT	2.59	3.08	4.05	5.79	0.67	0.77	1.31	2.1
9	MOCO + LiFT No Img.	4.58	8.78	13.01	15.48	1.18	2.69	4.95	7.27
10	MOCO + LiFT L1	6.12	9.80	13.73	14.98	1.59	3.22	5.86	7.53
11	MOCO + LiFT L2	6.37	10.08	13.91	**16.41**	1.53	3.12	5.99	**8.34**
12	MOCO + LiFT	**6.48**	**10.51**	**14.13**	16.34	**1.74**	**3.36**	**6.42**	8.05
13	ViT	1.26	5.72	13.23	16.9	0.27	1.62	4.89	7.34
14	ViT + Random LiFT	2.36	3.29	7.15	8.21	0.58	1.09	2.33	3.13
15	ViT + LiFT No Img.	2.94	7.76	15.69	18.74	0.79	2.22	5.68	8.23
16	ViT + LiFT L1	3.27	8.32	16.04	18.29	0.79	2.74	6.77	8.45
17	ViT + LiFT L2	3.57	8.78	16.29	**18.80**	0.97	2.64	**6.82**	**8.87**
18	ViT + LiFT	**3.76**	**9.21**	**16.5**8	18.69	**1.02**	**2.71**	6.63	8.81

three different backbones in this study: DINO [26], MoCo v3 (MoCo for short) [27], and a Fully Supervised ViT (ViT for short). We standardize the architecture to a ViT S/16 backbone for this analysis. We use the SPair-71k dataset and the keypoint correspondence task as the main representative metric for this analysis. The results are summarized in Table 8.9.

8.7.1.1 Random LiFT

One might question if LiFT actually benefits from training, or if the simple act of increasing the feature resolution with any arbitrary function is sufficient to improve performance. To test this question, we take a random initialization of the LiFT model and measure its performance. We denote this model as 'Random LiFT' in Table 8.9. It can clearly be seen in Table 8.9 rows 2, 8, and 14 that a randomly initialized LiFT model does not do anything meaningful as it performs poorly on all metrics. These results validate the importance of LiFT's self-supervised training

Table 8.10: Ablation of LiFT training epochs on ImageNet, including longer training. Results are shown for DINO+LiFT on Keypoint Correspondence using PCK@0.1.

Res/Epochs	5	10	30	50	100
112×112	17.47	17.53	17.75	17.97	18.14
224×224	28.45	28.50	28.65	29.00	29.11

method.

8.7.1.2 Ablation of Image Input to LiFT

In our approach, we increase the feature resolution through LiFT by also using the image as a source of finer spatial information. It should be noted that we use the image at the same resolution as was used to generate the initial features, which means LiFT does not have or require any additional information beyond the original ViT's input. To show the importance of this image information, we present a version of LiFT with the image input ablated, denoted as 'LiFT No Img.' in Table 8.9. We can see from rows 3 *vs.* 6, 9 *vs.* 12, and 15 *vs.* 18, that providing the image input helps LiFT produce better quality features which give improved performance on the keypoint correspondence task. It appears that ablating the image input is less harmful for higher-resolution inputs like 448, which makes intuitive sense as the feature map resolution is higher and thus more detail about the object boundaries can be represented. For DINO and the supervised ViT (rows 3 and 15), the no-image LiFT actually does very slightly better at 448 input resolution for PCK@0.1, but for all other cases normal LiFT is better. For PCK@0.05, the standard LiFT with image input is consistently much better. We believe this happens because LiFT can take direct cues regarding scene and object boundaries from the image input and generate higher resolution features which better respect these contours.

8.7.1.3 Effect of Distance Function

In our final approach, we use cosine distance to compute the loss between the ViT-generated higher resolution features and the upscaled features from LiFT. In Table 8.9, we compare with two alternative options for this distance function, specifically the L1 and L2 distance metrics. We denote these as 'LiFT L1' and 'LiFT L2' respectively. Cosine distance gives the best performance in most cases, such as in rows 4 & 5 *vs.* row 6, rows 10 & 11 *vs.* row 12, and rows 16 & 17 *vs.* row 18. For higher-resolution inputs, L2 distance is sometimes slightly better than cosine distance, but in most cases cosine is preferable. We believe this occurs because of the inherent normalization that cosine distance provides before computing the final loss.

8.7.1.4 Ablation of Training Epochs

As an additional experiment, we train the LiFT module on ImageNet for an extended period up to 100 epochs on 4 GPUs in Table 8.10. We find that there are small performance gains from training to very long epochs, however performance mostly saturates by epoch 5. At resolution 224, DINO+LiFT at 5 epochs gives a $\sim$3.7 point gain over the base DINO model, while training 95 epochs further only gives an additional 0.66 point gain. We believe this early saturation is thanks to the LiFT network's small size.

8.7.2 Additional Details for ViTDet+LiFT

For our experiments combining LiFT with ViTDet [339], we increase the size of our LiFT module to address the additional complexity of the task and backbone. To be consistent with ViTDet, we use an MAE-trained ViT-Base backbone instead of the ViT-Small used in our other

Table 8.11: Application of LiFT to various backbones for the Keypoint Correspondence task on SPair-71k for all metrics. LiFT gives consistent performance improvements.

	PCK@0.1				PCK@0.05				PCK@0.01			
Method/Resolution	56	112	224	448	56	112	224	448	56	112	224	448
DINO S/16	2.04	12.67	24.76	28.60	0.51	3.61	9.54	15.33	0.01	0.20	0.54	1.40
DINO S/16 + LiFT	5.05	17.72	28.68	31.38	1.19	6.29	14.72	18.90	0.06	0.29	0.91	2.52
DINO B/16	1.98	12.20	24.90	28.22	0.46	3.61	9.64	15.04	0.01	0.17	0.52	1.15
DINO B/16 + LiFT	5.43	17.74	29.35	31.27	1.29	6.56	14.80	18.10	0.04	0.37	0.92	2.43
DINO S/8	9.39	21.30	31.05	32.15	2.35	8.44	16.74	18.96	0.15	0.39	1.19	2.32
DINO S/8 + LiFT	12.90	26.73	34.54	34.58	4.35	11.99	20.61	21.01	0.18	0.75	2.21	3.77
DINO B/8	8.88	20.40	30.08	30.89	2.83	7.70	15.81	17.84	0.12	0.39	1.09	1.95
DINO B/8 + LiFT	12.21	25.17	33.23	33.17	4.22	11.73	19.39	20.18	0.13	0.69	2.27	3.32
MOCO S/16	1.27	3.43	7.37	12.31	0.21	0.84	2.35	5.49	0.00	0.03	0.10	0.31
MOCO S/16 + LiFT	6.48	10.51	14.13	16.34	1.74	3.36	6.42	8.05	0.04	0.16	0.42	0.73
ViT S/16	1.26	5.72	13.23	16.90	0.27	1.62	4.89	7.34	0.02	0.06	0.30	0.50
ViT S/16 + LiFT	3.76	9.21	16.58	18.69	1.02	2.71	6.63	8.81	0.02	0.13	0.45	0.72
Leopart S/16	2.35	11.20	23.33	26.54	0.60	3.22	8.90	12.26	0.05	0.10	0.47	0.79
Leopart S/16 + LiFT	4.24	15.61	27.77	30.06	1.22	5.16	12.81	15.66	0.02	0.25	0.74	1.39

primary experiments. Note that a standard ViT-Small model outputs feature maps with 384 channels, while ViT-Base outputs 768 channels. To handle the increased number of channels, we commensurately increase the number of channels in the layers of our LiFT module. We also add an additional convolutional block to the encoder segment. This larger LiFT module has a total of $7M$ parameters, as compared with the $1.2M$ parameter version used for smaller architectures. The ViTDet model used has $111M$ parameters, so our combined ViTDet+LiFT architecture has $118M$ parameters total. This is a 6.3% increase in total parameters, which is similar to the relative percentage increase of the smaller LiFT version used with DINO S/16. Also, here we train LiFT on the COCO dataset in place of ImageNet. Because the COCO dataset is much smaller than ImageNet, we train on it for 100 epochs.

8.7.3 Additional Backbones with LiFT

8.7.3.1 Performance Improvements with LiFT for Other Backbones

We further demonstrate the general utility of LiFT by applying it to several additional backbones, including Leopart [337] and several other DINO [26] ViTs, namely ViT-S/8, ViT-B/16, and ViT-B/8. The results are summarized in Table 8.11. LiFT shows consistent improvement for the various architectures and models across patch sizes (8 and 16), trainings (Leopart and DINO) and backbone sizes (Base and Small).

8.7.3.2 Cost *vs*. Performance Trade-off for Other Backbones

We extend the Performance *vs*. Compute Cost analysis from Section 8.4.2 to include both the MOCO and fully-supervised ViT backbones, as shown in Figure 8.6. We find that LiFT consistently boosts the performance of all three backbones at all FLOP allowances.

8.7.4 Additional Metrics

We present results for additional metrics on the SPair-71k [72] and DAVIS [71] datasets. For SPair-71k, we additionally report the values for PCK@0.01 in Table 8.12. For DAVIS, we also report the J Mean and F Mean in Table 8.13. We present these results alongside the previously reported metrics for completeness. For SPair-71k, the PCK@0.01 metric demands the most precision, and the additional feature resolution provided by LiFT provides consistent improvements. For DAVIS, both the J Mean and F Mean are also consistently improved by adding LiFT.

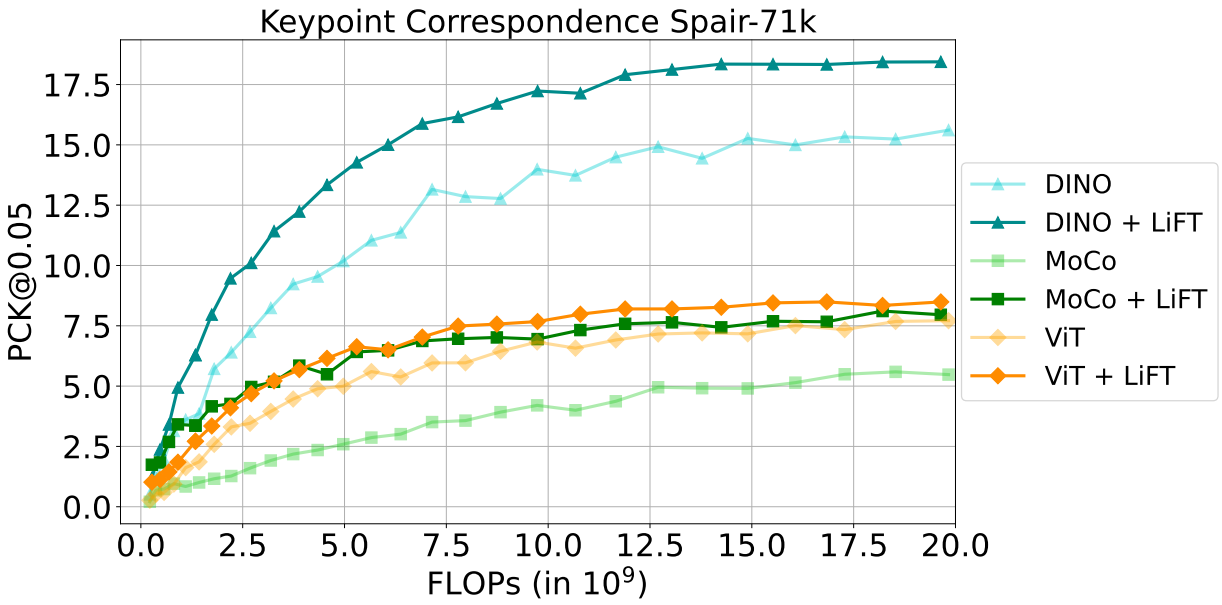


Figure 8.6: Performance *vs.* Compute Cost trade-off curve for LiFT when combined with different ViT backbones. Results are presented for SPair-71k Keypoint Correspondence. LiFT provides a performance boost for all three backbones at any FLOP budget.

8.7.5 Additional Similarity Map Samples

In Section 8.5.2, we found that the feature self-similarity maps for DINO+LiFT more clearly and sharply outline the central object in an image. To further highlight this, we provide additional samples further showing the benefits of LiFT for self-similarity maps, as shown in Figure 8.7. In rows 1 to 8 (left), we show samples with single central objects of differing shapes and sizes. We see that the feature self-similarity maps for DINO+LiFT more uniformly fill the foreground object region, and have less noisy correlations with background regions. In rows 1 to 3 (right), we show samples where the central feature vector, shown by the red marker, lies on a background region. In these cases, we still see sharp contours around the foreground objects, or around the body of water in row 1 (right). In cases like rows 4 to 8 (right), when there are multiple overlapping instances of the same object class, we see a uniform highlighting

Table 8.12: Comparison between LiFT and other baselines on the Keypoint Correspondence task on SPair-71k for all metrics.

	PCK@0.1				PCK@0.05				PCK@0.01			
Method/Resolution	56	112	224	448	56	112	224	448	56	112	224	448
DINO	2.04	12.67	24.76	28.6	0.51	3.61	9.54	15.33	0.01	0.2	0.54	1.4
Leopart	2.35	11.2	23.33	26.54	0.6	3.22	8.9	12.26	0.05	0.1	0.47	0.79
SelfPatch	2.13	12.18	23.03	27.34	0.44	3.61	9.32	14.44	0.02	0.17	0.42	1.12
DINO+LiFT	**5.05**	**17.72**	**28.68**	**31.38**	**1.19**	**6.29**	**14.72**	**18.90**	**0.06**	**0.29**	**0.91**	**2.52**

Table 8.13: Comparison between LiFT and other baselines on the DAVIS Video Object Segmentation task with additional metrics J Mean and F Mean.

	J Mean				F Mean				J & F Mean			
Method/Resolution	56	112	224	448	56	112	224	448	56	112	224	448
DINO	0.10	0.22	0.38	0.52	0.05	0.13	0.28	0.50	0.07	0.18	0.33	0.51
DINO+Bilinear	0.14	0.29	0.43	0.54	0.08	0.18	0.31	0.52	0.11	0.24	0.37	0.53
Leopart	0.09	0.20	0.35	0.47	0.05	0.12	0.26	0.43	0.07	0.16	0.30	0.45
SelfPatch	0.10	0.22	0.38	0.53	0.05	0.13	0.28	0.50	0.07	0.17	0.33	0.51
DINO+LiFT	**0.16**	**0.33**	**0.48**	**0.59**	**0.10**	**0.23**	**0.41**	**0.63**	**0.13**	**0.28**	**0.44**	**0.61**

of the multiple object instances. We also see that DINO+LiFT better highlights thin structures in objects, like the teapot handle and tripod legs in rows 7 and 8 (left). For comparison, when DINO (without LiFT) is given a doubled input size, these details are sometimes lost to noisy background regions.

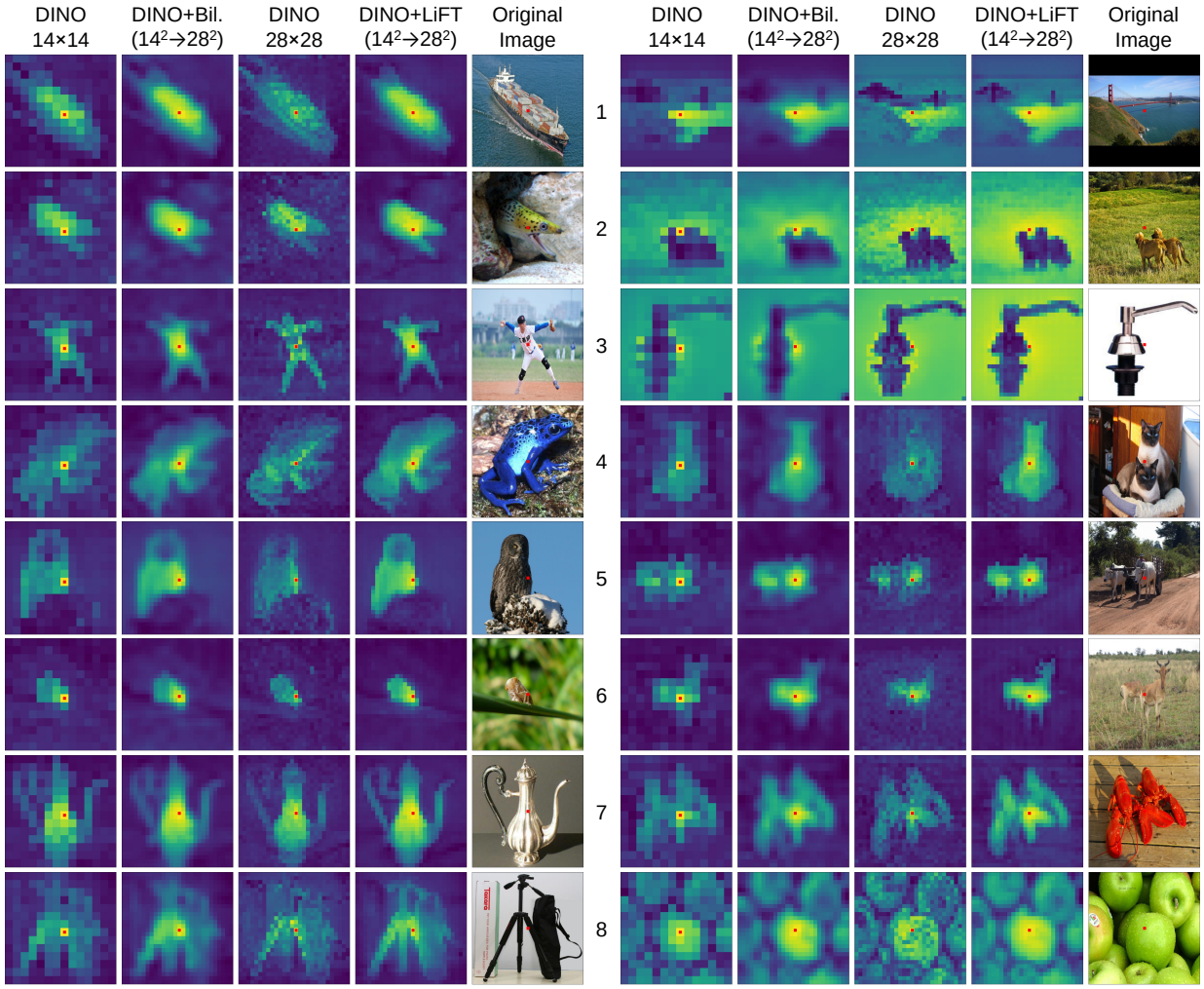


Figure 8.7: Additional visualizations of the self-similarity of features extracted from DINO, DINO+Bilinear interpolation, DINO with higher resolution image, and DINO+LiFT. The input image is shown for comparison. The self-similarity is computed using the feature corresponding to the center of the grid (marked in red) and all other features from each spatial location. Brighter pixels show a higher similarity.

Chapter 9: Future Work and Conclusion

9.1 Future Work

There are a few possible research directions which build up on the work I have done which I am excited about. For future work, we plan to continue research and exploration along the following directions and discuss them in more detail in the next sections.

9.1.1 Feedback Guided Synthetic Data Generation

In Gen2Det [368] we started by generating a large number of synthetic images without worrying about sample complexity or its actual usefulness to training. As shown in Figure 9.1, the next step in such an approach to better use synthetic data is to generate few but high impact samples which help the model learn better. This will not only reduce the training time, as instead of training on thousands of generated images we would generate a fewer useful images, it would also make the generation faster as a whole due to the sheer reduction in number of generated instances.

To perform this feedback guided sampling which follows the essence of hard sample mining, we propose two techniques. The first is a simpler offline technique, where a pretrained detector which has been trained only on the real data provides direct signal whether a sample is

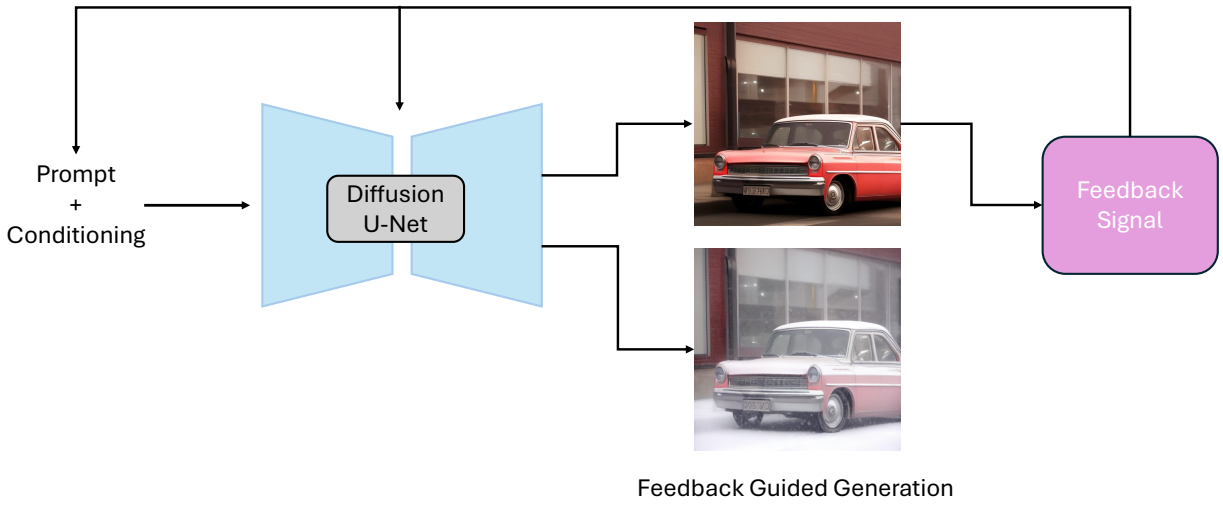


Figure 9.1: Utilizing feedback while generating images to generate useful images for downstream training.

easy or hard based on the prediction logits. Using this detector which is frozen we can guide the diffusion model to generate samples which incur a higher loss. In doing so we would still need to ensure that the sample quality and correctness is maintained using auxialiary losses. Another way to go about this is using the detector being trained as a feedback mechanism. This approach is more adaptive as it takes into account the sample hardness with respect to the current state of the detector model being trained. While this seems more suitable, this is an online approach and hence would incur generation cost during training making the training slower. We are confident as the diffusion sampling becomes more efficient, even this approach would become more viable.

9.1.2 Memory Augmented Multimodal Learning

After exploring different supervision strategies and data settings, it is clear than training a model which continuously improve and adapt to the ever changing world is a hard task. While training with large amounts of data and trying to learn all possible concepts and their compositions is an important research direction, another way to train models which is effective is to

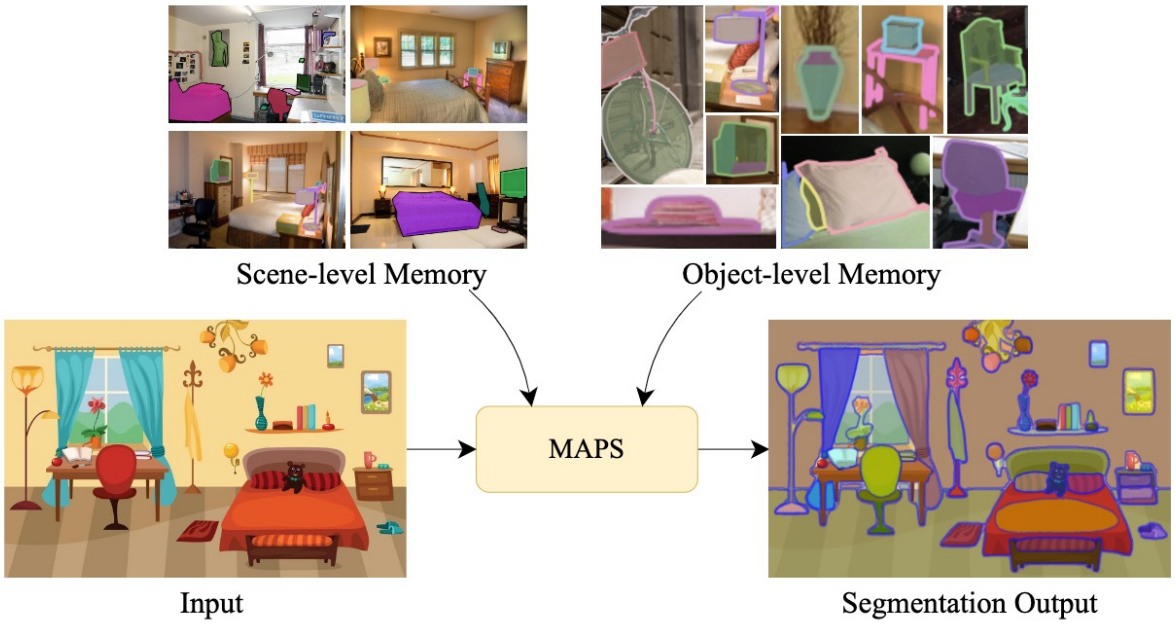


Figure 9.2: Utilizing memory for multimodal learning in this case for memory augmented panoptic segmentation (MAPS).

utilize memory. Similar to humans effectively utilizing their memory to understand and perform tasks, models having such capabilities are also getting popular [369, 370, 371, 372, 373].

We propose a general purpose model which utilizes memory in a scalable and modular way. We propose to start with the task of panoptic segmentation which is an important vision tasks that requires good understanding of pixel level content. Our memory augmented panoptic segmentation (MAPS) would be easy to integrate into any transformer based architecture with minor changes. Additionally, having such a memory enables the model to stay updated without changing the weights. Rather one could grow and change the memory once the model is trained to introduce new concepts. Finally, such an approach makes the models multimodal as we can use multimodal models to encode the data and generate the memory making these models more general and able to take advantage of data from multiple modalities.

9.1.3 Unified Techniques for Data Processing

Current techniques in computer vision are very data and modality specific. While this leads to good models for the specific domain/modality as input/output, it does not have the flexibility to generalize across modalities. Vision Language Models (VLMs) like CLIP [62, 374, 375] have bridged the gap when it comes to designing more general models for image and text by providing a good representation space where semantics are aligned well between the two modalities. While this is a first step in a direction of utilizing unified representation, we can take a bigger leap by considering the new generation of similar models which are truly multimodal in the sense that they align multiple modalities. Models like ImageBind [376], UniBind [377] and OmniBind [378] give us more flexibility by aligning multiple modalities like audio, text, image, video, point clouds etc. By using such models we can design approaches which are general in terms of data processing, filtering and hard mining. Further, such encoders with a shared multimodal space can also act as strong conditioning agents. Where unlike current generators that utilize only one or two kinds of conditioning, by using these multimodal encoders we can extend conditioning inputs to various different modalities.

9.2 Conclusion

Through this thesis we have looked at various supervision and data strategies commonly used across recognition and generation landscapes. Starting with Teaching Matters [335] we look at the role of supervision across different self-supervised and fully-supervised ViT backbones. We dive deep into the attention and feature representation to understand the changes that occur when the same ViT backbone is trained with variations in training losses, recipes and data.

We then dive deep into specific supervision scenarios tackling the problem of sparse supervision for object detection through our proposed method SparseDet [379] where we train object detectors with missing labels. By utilizing a mix of semi and self-supervised losses we are able to utilize sparse annotation better and reduce the negative effect of missing annotation regions being labeled as background. Finally, on the recognition front we also look at open world discovery style framework for synthetic image source attribution [380]. In this work we propose a module pipeline to adaptively learn and cluster generated images coming from the same source into the same cluster without any need for additional information regarding number of classes. Our method is able to handle sources unseen during training and also works in an online setting.

On the generative modeling end we first look at highlighting issues with existing image-to-image translation works in our work GRiT [381]. We show how existing losses are competing against each other and propose a decoupled approach to generating a reconstruction and residual image. By decoupling the outputs we are able to achieve better realism and show better outputs. Moving on from images we also propose two works for generating videos. The first work LSR [382] uses a decoupled training paradigm to generate talking head video using few shots of a new subject. Diff2Lip [383] on the other hand incorporates audio in addition to video to generate high quality lip-synced talking heads.

Finally, given our learning from both generation and recognition domains we identify and propose two new works which utilize generation to improve recognition. To this end we present Gen2Det [368] where we utilize synthetic data from grounded diffusion models to improve object detection and segmentation performance. We present a modular pipeline which is able to generate, filter and utilize synthetic data effectively and show improved performance especially in low data and long tailed settings. Following up on our ViT analysis work we also present LiFT [384]

which overcomes the issue of low resolution feature maps for dense task. Through LiFT we propose a lightweight network which is easy to train and use and incurs minimal computational cost but is able to provide high resolution features which are much more useful for dense downstream tasks.

While these works cover a wide breadth of supervision and data paradigms accross recognition and generation, we in no way touch all existing directions but are positive our insights would be helpful for future research along the directions we explored.

Bibliography

[1] 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, .

[2] Ning Yu, Larry S Davis, and Mario Fritz. Attributing fake images to gans: Learning and analyzing gan fingerprints. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7556–7566, 2019.

[3] Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A Efros. Cnn-generated images are surprisingly easy to spot... for now. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 7, 2020.

[4] Kai Han, Sylvestre-Alvise Rebuffi, Sebastien Ehrhardt, Andrea Vedaldi, and Andrew Zisserman. Automatically discovering and learning new visual categories with ranking statistics. *arXiv preprint arXiv:2002.05714*, 2020.

[5] Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.

[6] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman. Voxceleb2: Deep speaker recognition. *arXiv preprint arXiv:1806.05622*, 2018.

[7] Joon Son Chung and Andrew Zisserman. Lip reading in the wild. In Shang-Hong Lai, Vincent Lepetit, Ko Nishino, and Yoichi Sato, editors, *Computer Vision – ACCV 2016*, pages 87–103, Cham, 2017. Springer International Publishing. ISBN 978-3-319-54184-6.

[8] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *CVPR*, 2019.

[9] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *CVPR*, 2019.

[10] Konstantinos Vougioukas, Stavros Petridis, and Maja Pantic. Dino: A conditional energy-based gan for domain translation. *ICLR*, 2021.

[11] Fangneng Zhan, Jiahui Zhang, Yingchen Yu, Rongliang Wu, and Shijian Lu. Modulated contrast for versatile image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18280–18290, 2022.

[12] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *CVPR*, 2018.

[13] KR Prajwal, Rudrabha Mukhopadhyay, Vinay P Namboodiri, and CV Jawahar. A lip sync expert is all you need for speech to lip generation in the wild. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 484–492, 2020.

[14] Hang Zhou, Yasheng Sun, Wayne Wu, Chen Change Loy, Xiaogang Wang, and Ziwei Liu. Pose-controllable talking face generation by implicitly modularized audio-visual representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[15] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[16] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? *Advances in Neural Information Processing Systems*, 34:12116–12128, 2021.

[17] Kristen Grauman and Trevor Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1458–1465. IEEE, 2005.

[18] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, volume 2, pages 2169–2178. IEEE, 2006.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015.

[20] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[21] Abhinav Shrivastava, Rahul Sukthankar, Jitendra Malik, and Abhinav Gupta. Beyond skip connections: Top-down modulation for object detection. *arXiv preprint arXiv:1612.06851*, 2016.

[22] Pedro O Pinheiro, Tsung-Yi Lin, Ronan Collobert, and Piotr Dollár. Learning to refine object segments. In *European conference on computer vision*, pages 75–91. Springer, 2016.

[23] Chunshui Cao, Xianming Liu, Yi Yang, Yinan Yu, Jiang Wang, Zilei Wang, Yongzhen Huang, Liang Wang, Chang Huang, Wei Xu, et al. Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2956–2964, 2015.

[24] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6541–6549, 2017.

[25] Jesse Mu and Jacob Andreas. Compositional explanations of neurons. *Advances in Neural Information Processing Systems*, 33:17153–17163, 2020.

[26] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[27] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[28] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022.

[29] Hangbo Bao, Li Dong, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021.

[30] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[31] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, pages 12310–12320. PMLR, 2021.

[32] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. *Advances in neural information processing systems*, 32, 2019.

[33] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

[34] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

[35] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6707–6717, 2020.

[36] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

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

[38] Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, and Luc V Gool. Revisiting contrastive methods for unsupervised learning of visual representations. *Advances in Neural Information Processing Systems*, 34:16238–16250, 2021.

[39] Elijah Cole, Xuan Yang, Kimberly Wilber, Oisin Mac Aodha, and Serge Belongie. When does contrastive visual representation learning work? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14755–14764, 2022.

[40] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning*, pages 9929–9939. PMLR, 2020.

[41] Feng Wang and Huaping Liu. Understanding the behaviour of contrastive loss. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2495–2504, 2021.

[42] Senthil Purushwalkam and Abhinav Gupta. Demystifying contrastive self-supervised learning: Invariances, augmentations and dataset biases. *Advances in Neural Information Processing Systems*, 33:3407–3418, 2020.

[43] Grant Van Horn, Elijah Cole, Sara Beery, Kimberly Wilber, Serge Belongie, and Oisin Mac Aodha. Benchmarking representation learning for natural world image collections. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12884–12893, 2021.

[44] Klemen Kotar, Gabriel Ilharco, Ludwig Schmidt, Kiana Ehsani, and Roozbeh Mottaghi. Contrasting contrastive self-supervised representation learning pipelines. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9949–9959, 2021.

[45] Tom George Grigg, Dan Busbridge, Jason Ramapuram, and Russ Webb. Do self-supervised and supervised methods learn similar visual representations? *arXiv preprint arXiv:2110.00528*, 2021.

[46] Matthew Gwilliam and Abhinav Shrivastava. Beyond supervised vs. unsupervised: Representative benchmarking and analysis of image representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9642–9652, 2022.

[47] Sayak Paul and Pin-Yu Chen. Vision transformers are robust learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 2071–2081, 2022.

[48] Rulin Shao, Zhouxing Shi, Jinfeng Yi, Pin-Yu Chen, and Cho-Jui Hsieh. On the adversarial robustness of vision transformers. *arXiv preprint arXiv:2103.15670*, 2021.

[49] Daquan Zhou, Zhiding Yu, Enze Xie, Chaowei Xiao, Animashree Anandkumar, Jiashi Feng, and Jose M Alvarez. Understanding the robustness in vision transformers. In *International Conference on Machine Learning*, pages 27378–27394. PMLR, 2022.

[50] Yanghao Li, Saining Xie, Xinlei Chen, Piotr Dollar, Kaiming He, and Ross Girshick. Benchmarking detection transfer learning with vision transformers. *arXiv preprint arXiv:2111.11429*, 2021.

[51] Shuhao Cao, Peng Xu, and David A Clifton. How to understand masked autoencoders. *arXiv preprint arXiv:2202.03670*, 2022.

[52] Arda Sahiner, Tolga Ergen, Batu Ozturkler, John Pauly, Morteza Mardani, and Mert Pilanci. Unraveling attention via convex duality: Analysis and interpretations of vision transformers. *arXiv preprint arXiv:2205.08078*, 2022.

[53] Hugo Touvron, Matthieu Cord, Alaaeldin El-Nouby, Jakob Verbeek, and Hervé Jégou. Three things everyone should know about vision transformers. *arXiv preprint arXiv:2203.09795*, 2022.

[54] Namuk Park and Songkuk Kim. How do vision transformers work? *arXiv preprint arXiv:2202.06709*, 2022.

[55] Shir Amir, Yossi Gandelsman, Shai Bagon, and Tali Dekel. Deep vit features as dense visual descriptors. *ECCVW What is Motion For?*, 2022.

[56] Mehmet Aygün and Oisin Mac Aodha. Demystifying unsupervised semantic correspondence estimation. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXX*, pages 125–142. Springer, 2022.

[57] Amin Ghiasi, Hamid Kazemi, Eitan Borgnia, Steven Reich, Manli Shu, Micah Goldblum, Andrew Gordon Wilson, and Tom Goldstein. What do vision transformers learn? a visual exploration. *arXiv preprint arXiv:2212.06727*, 2022.

[58] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[59] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

[60] Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your vit? data, augmentation, and regularization in vision transformers. *arXiv preprint arXiv:2106.10270*, 2021.

[61] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019.

[62] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[63] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3733–3742, 2018.

[64] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9640–9649, 2021.

[65] Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Scan: Learning to classify images without labels. In *European conference on computer vision*, pages 268–285. Springer, 2020.

[66] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[67] Ju He, Shuo Yang, Shaokang Yang, Adam Kortylewski, Xiaoding Yuan, Jie-Neng Chen, Shuai Liu, Cheng Yang, and Alan Yuille. Partimagenet: A large, high-quality dataset of parts. *arXiv preprint arXiv:2112.00933*, 2021.

[68] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[69] Filip Radenović, Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondřej Chum. Revisiting oxford and paris: Large-scale image retrieval benchmarking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5706–5715, 2018.

[70] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2008.

[71] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017.

[72] Juhong Min, Jongmin Lee, Jean Ponce, and Minsu Cho. Spair-71k: A large-scale benchmark for semantic correspondence. *arXiv preprint arXiv:1908.10543*, 2019.

[73] Mark Hamilton, Zhoutong Zhang, Bharath Hariharan, Noah Snavely, and William T Freeman. Unsupervised semantic segmentation by distilling feature correspondences. *arXiv preprint arXiv:2203.08414*, 2022.

[74] 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.

[75] 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. *arXiv preprint arXiv:2109.14279*, 2021.

[76] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Algorithms for learning kernels based on centered alignment. *The Journal of Machine Learning Research*, 13:795–828, 2012.

[77] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pages 3519–3529. PMLR, 2019.

[78] Thao Nguyen, Maithra Raghu, and Simon Kornblith. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. *arXiv preprint arXiv:2010.15327*, 2020.

[79] Allan Jabri, Andrew Owens, and Alexei Efros. Space-time correspondence as a contrastive random walk. *Advances in neural information processing systems*, 33:19545–19560, 2020.

[80] Yi Yang and Deva Ramanan. Articulated human detection with flexible mixtures of parts. *IEEE transactions on pattern analysis and machine intelligence*, 35(12):2878–2890, 2012.

[81] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 702–703, 2020.

[82] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[83] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[84] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

[85] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[86] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[87] Anand Subramanian. Torch cka. https://github.com/AntixK/PyTorch-Model-Compare, 2021.

[88] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016.

[89] Kun Xu, Yong Li, Tao Ju, Shi-Min Hu, and Tian-Qiang Liu. Efficient affinity-based edit propagation using kd tree. *ACM Transactions on Graphics (TOG)*, 28(5):1–6, 2009.

[90] Simon Chadwick and Paul Newman. Training object detectors with noisy data. *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1319–1325, 2019.

[91] Zhe Wu, Navaneeth Bodla, Bharat Singh, Mahyar Najibi, Rama Chellappa, and Larry S. Davis. Soft sampling for robust object detection. In *BMVC*, 2019.

[92] Junnan Li, Caiming Xiong, Richard Socher, and Steven C. H. Hoi. Towards noise-resistant object detection with noisy annotations. *ArXiv*, abs/2003.01285, 2020.

[93] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, K. Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. URL https://api.semanticscholar.org/CorpusID:57246310.

[94] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014.

[95] Alina Kuznetsova, Hassan Rom, Neil Gordon Alldrin, Jasper R. R. Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, Tom Duerig, and Vittorio Ferrari. The open images dataset v4. *International Journal of Computer Vision*, 128:1956–1981, 2020.

[96] Yusuke Niitani, Takuya Akiba, Tommi Kerola, Toru Ogawa, Shotaro Sano, and Shuji Suzuki. Sampling techniques for large-scale object detection from sparsely annotated objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6510–6518, 2019.

[97] Han Zhang, Fangyi Chen, Zhiqiang Shen, Qiqi Hao, Chenchen Zhu, and Marios Savvides. Solving missing-annotation object detection with background recalibration loss. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1888–1892. IEEE, 2020.

[98] Yuewei Yang, Kevin J Liang, and Lawrence Carin. Object detection as a positive-unlabeled problem. In *BMVC*, 2020.

[99] Tiancai Wang, Tong Yang, Jiale Cao, and X. Zhang. Co-mining: Self-supervised learning for sparsely annotated object detection. In *AAAI*, 2021.

[100] 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, .

[101] Ke Yan, Jinzheng Cai, Adam P. Harrison, Dakai Jin, Jing Xiao, and Le Lu. Universal lesion detection by learning from multiple heterogeneously labeled datasets. *ArXiv*, abs/2005.13753, 2020.

[102] Hansheng Li, Xin Han, Yuxin Kang, Xiaoshuang Shi, Mengdi Yan, Zixu Tong, Qirong Bu, Lei Cui, Jun Feng, and Lin Yang. A novel loss calibration strategy for object detection networks training on sparsely annotated pathological datasets. In *MICCAI*, 2020.

[103] Yongqiang Zhang, Mingli Ding, Yancheng Bai, Mengmeng Xu, and Bernard Ghanem. Beyond weakly supervised: Pseudo ground truths mining for missing bounding-boxes object detection. *IEEE Transactions on Circuits and Systems for Video Technology*, 30: 983–997, 2020.

[104] Ke Yan, Jinzheng Cai, Youjing Zheng, Adam P. Harrison, Dakai Jin, You-Bao Tang, Yuxing Tang, Lingyun Huang, Jing Xiao, and Le Lu. Learning from multiple datasets with heterogeneous and partial labels for universal lesion detection in ct. *IEEE Transactions on Medical Imaging*, 40:2759–2770, 2021.

[105] Jisoo Jeong, Seungeui Lee, Jeesoo Kim, and Nojun Kwak. Consistency-based semi-supervised learning for object detection. *Advances in neural information processing systems*, 32:10759–10768, 2019.

[106] Peng Tang, Chetan Ramaiah, Yan Wang, Ran Xu, and Caiming Xiong. Proposal learning for semi-supervised object detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2291–2301, 2021.

[107] Kihyuk Sohn, Zizhao Zhang, Chun-Liang Li, Han Zhang, Chen-Yu Lee, and Tomas Pfister. A simple semi-supervised learning framework for object detection. *arXiv preprint arXiv:2005.04757*, 2020.

[108] Yen-Cheng Liu, Chih-Yao Ma, Zijian He, Chia-Wen Kuo, Kan Chen, Peizhao Zhang, Bichen Wu, Zsolt Kira, and Peter Vajda. Unbiased teacher for semi-supervised object detection. *arXiv preprint arXiv:2102.09480*, 2021.

[109] Mengde Xu, Zheng Zhang, Han Hu, Jianfeng Wang, Lijuan Wang, Fangyun Wei, Xiang Bai, and Zicheng Liu. End-to-end semi-supervised object detection with soft teacher. *arXiv preprint arXiv:2106.09018*, 2021.

[110] Qize Yang, Xihan Wei, Biao Wang, Xian-Sheng Hua, and Lei Zhang. Interactive self-training with mean teachers for semi-supervised object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5941–5950, 2021.

[111] Jihun Yoon, Seungbum Hong, and Min-Kook Choi. Semi-supervised object detection with sparsely annotated dataset. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 719–723. IEEE, 2021.

[112] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[113] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[114] Ross Girshick. Fast r-cnn. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pages 1440–1448, Washington, DC, USA, 2015. IEEE Computer Society. ISBN 978-1-4673-8391-2. doi: 10.1109/ICCV.2015.169. URL http://dx.doi.org/10.1109/ICCV.2015.169.

[115] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[116] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:1137–1149, 2015.

[117] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *NIPS*, pages 379–387, 2016. URL http://dblp.uni-trier.de/db/conf/nips/nips2016.html#DaiLHS16.

[118] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, 2017.

[119] Bharat Singh and Larry S Davis. An analysis of scale invariance in object detection-snip. *CVPR*, 2018.

[120] Bharat Singh, Mahyar Najibi, and Larry S Davis. SNIPER: Efficient multi-scale training. *NeurIPS*, 2018.

[121] Mahyar Najibi, Bharat Singh, and Larry S Davis. AutoFocus: Efficient multi-scale inference. *ICCV*, 2019.

[122] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017. doi: 10.1109/ICCV.2017.322.

[123] Alexander Kirillov, Ross B. Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *CVPR*, pages 6399–6408. Computer Vision Foundation / IEEE, 2019.

[124] Alexander Kirillov, Kaiming He, Ross B. Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. In *CVPR*, pages 9404–9413. Computer Vision Foundation / IEEE, 2019.

[125] Justin Johnson Georgia Gkioxari, Jitendra Malik. Mesh r-cnn. In *ICCV*, 2019.

[126] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 21–37. Springer, 2016.

[127] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007, 2017.

[128] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2015. URL `http://arxiv.org/abs/1506.02640`. cite arxiv:1506.02640.

[129] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9627–9636, 2019.

[130] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[131] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42:386–397, 2020.

[132] Ross B. Girshick. Fast R-CNN. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.

[133] 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)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

[134] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[135] Hanjun Li, Xingjia Pan, Ke Yan, Fan Tang, and Wei-Shi Zheng. Siod: Single instance annotated per category per image for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14197–14206, 2022.

[136] Akshay Raj Dhamija, Manuel Günther, Jonathan Ventura, and Terrance E. Boult. The overlooked elephant of object detection: Open set. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1010–1019, 2020. doi: 10.1109/WACV45572.2020.9093355.

[137] Huy V. Vo, Patrick Pérez, and Jean Ponce. Toward unsupervised, multi-object discovery in large-scale image collections. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[138] Huy V. Vo, Francis 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, 2019.

[139] Sai Saketh Rambhatla, Rama Chellappa, and Abhinav Shrivastava. The pursuit of knowledge: Discovering and localizing novel categories using dual memory. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[140] Sai Saketh Rambhatla, Ishan Misra, Rama Chellappa, and Abhinav Shrivastava. The pursuit of knowledge: Discovering and localizing novel categories using dual memory. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2023.

[141] Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. Fake news detection on social media: A data mining perspective. *ACM SIGKDD explorations newsletter*, 19(1): 22–36, 2017.

[142] Andreas Rössler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. Faceforensics: A large-scale video dataset for forgery detection in human faces. *arXiv preprint arXiv:1803.09179*, 2018.

[143] Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. Faceforensics++: Learning to detect manipulated facial images. In *ICCV*, pages 1–11, 2019.

[144] Francesco Marra, Diego Gragnaniello, Davide Cozzolino, and Luisa Verdoliva. Detection of gan-generated fake images over social networks. In *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 384–389. IEEE, 2018.

[145] Xu Zhang, Svebor Karaman, and Shih-Fu Chang. Detecting and simulating artifacts in gan fake images. In *2019 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6. IEEE, 2019.

[146] F. Marra, D. Gragnaniello, L. Verdoliva, and G. Poggi. Do gans leave artificial fingerprints? In *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 506–511, 2019. doi: 10.1109/MIPR.2019.00103.

[147] Michael Albright and Scott McCloskey. Source generator attribution via inversion. *arXiv preprint arXiv:1905.02259*, 2019.

[148] Jay Yagnik, Dennis Strelow, David A Ross, and Ruei-sung Lin. The power of comparative reasoning. In *2011 International Conference on Computer Vision*, pages 2431–2438. IEEE, 2011.

[149] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems*, pages 7167–7177, 2018.

[150] Yen-Chang Hsu, Yilin Shen, Hongxia Jin, and Zsolt Kira. Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10951–10960, 2020.

[151] Walter J Scheirer, Anderson de Rezende Rocha, Archana Sapkota, and Terrance E Boult. Toward open set recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(7):1757–1772, 2012.

[152] Lalit P Jain, Walter J Scheirer, and Terrance E Boult. Multi-class open set recognition using probability of inclusion. In *European Conference on Computer Vision*, pages 393–409. Springer, 2014.

[153] Poojan Oza and Vishal M Patel. C2ae: Class conditioned auto-encoder for open-set recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2307–2316, 2019.

[154] Ethan M Rudd, Lalit P Jain, Walter J Scheirer, and Terrance E Boult. The extreme value machine. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):762–768, 2017.

[155] Terrance E Boult, Steve Cruz, Akshay Raj Dhamija, M Gunther, James Henrydoss, and Walter J Scheirer. Learning and the unknown: Surveying steps toward open world recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9801–9807, 2019.

[156] Abhijit Bendale and Terrance Boult. Towards open world recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1893–1902, 2015.

[157] Hu Xu, Bing Liu, Lei Shu, and P Yu. Open-world learning and application to product classification. In *The World Wide Web Conference*, pages 3413–3419, 2019.

[158] Yen-Chang Hsu, Zhaoyang Lv, and Zsolt Kira. Learning to cluster in order to transfer across domains and tasks. *arXiv preprint arXiv:1711.10125*, 2017.

[159] Kai Han, Andrea Vedaldi, and Andrew Zisserman. Learning to discover novel visual categories via deep transfer clustering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8401–8409, 2019.

[160] Zifeng Wang, Batool Salehi, Andrey Gritsenko, Kaushik Chowdhury, Stratis Ioannidis, and Jennifer Dy. Open-world class discovery with kernel networks. *arXiv preprint arXiv:2012.06957*, 2020.

[161] Thomas Dean, Mark A Ruzon, Mark Segal, Jonathon Shlens, Sudheendra Vijayanarasimhan, and Jay Yagnik. Fast, accurate detection of 100,000 object classes on a single machine. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1814–1821, 2013.

[162] Saquib Sarfraz, Vivek Sharma, and Rainer Stiefelhagen. Efficient parameter-free clustering using first neighbor relations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8934–8943, 2019.

[163] Xueting Yan, Ishan Misra, Abhinav Gupta, Deepti Ghadiyaram, and Dhruv Mahajan. Clusterfit: Improving generalization of visual representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6509–6518, 2020.

[164] Xu Yang, Cheng Deng, Feng Zheng, Junchi Yan, and Wei Liu. Deep spectral clustering using dual autoencoder network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4066–4075, 2019.

[165] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856, 2002.

[166] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4): 395–416, 2007.

[167] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5147–5156, 2016.

[168] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487, 2016.

[169] Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. Improved deep embedded clustering with local structure preservation. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 1753–1759, 2017. doi: 10.24963/ijcai.2017/243. URL `https://doi.org/10.24963/ijcai.2017/243`.

[170] Yazhou Ren, Kangrong Hu, Xinyi Dai, Lili Pan, Steven CH Hoi, and Zenglin Xu. Semi-supervised deep embedded clustering. *Neurocomputing*, 325:121–130, 2019.

[171] Lei Yang, Xiaohang Zhan, Dapeng Chen, Junjie Yan, Chen Change Loy, and Dahua Lin. Learning to cluster faces on an affinity graph. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2298–2306, 2019.

[172] Francesco Marra, Cristiano Saltori, Giulia Boato, and Luisa Verdoliva. Incremental learning for the detection and classification of gan-generated images. In *2019 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6. IEEE, 2019.

[173] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[174] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8789–8797, 2018.

[175] Zhenliang He, Wangmeng Zuo, Meina Kan, Shiguang Shan, and Xilin Chen. Attgan: Facial attribute editing by only changing what you want. *IEEE Transactions on Image Processing*, 28(11):5464–5478, 2019.

[176] David Berthelot, Thomas Schumm, and Luke Metz. Began: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017.

[177] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[178] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[179] Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. Mmd gan: Towards deeper understanding of moment matching network. In *Advances in Neural Information Processing Systems*, pages 2203–2213, 2017.

[180] Marc G Bellemare, Ivo Danihelka, Will Dabney, Shakir Mohamed, Balaji Lakshminarayanan, Stephan Hoyer, and Rémi Munos. The cramer distance as a solution to biased wasserstein gradients. *arXiv preprint arXiv:1705.10743*, 2017.

[181] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[182] Karol Kurach, Mario Lučić, Xiaohua Zhai, Marcin Michalski, and Sylvain Gelly. A large-scale study on regularization and normalization in gans. In *International Conference on Machine Learning*, pages 3581–3590. PMLR, 2019.

[183] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[184] Mario Lucic, Michael Tschannen, Marvin Ritter, Xiaohua Zhai, Olivier Bachem, and Sylvain Gelly. High-fidelity image generation with fewer labels. *arXiv preprint arXiv:1903.02271*, 2019.

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

[186] Tomasz Malisiewicz, Abhinav Gupta, and Alexei A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *ICCV*, 2011.

[187] Abhinav Shrivastava, Tomasz Malisiewicz, Abhinav Gupta, and Alexei A. Efros. "data-driven visual similarity for cross-domain image matching". *"ACM Transaction of Graphics (TOG) (Proceedings of ACM SIGGRAPH ASIA)"*, 30(6), 2011.

[188] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[189] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006.

[190] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[191] Negar Rostamzadeh, Seyedarian Hosseini, Thomas Boquet, Wojciech Stokowiec, Ying Zhang, Christian Jauvin, and Chris Pal. Fashion-gen: The generative fashion dataset and challenge. *arXiv preprint arXiv:1806.08317*, 2018.

[192] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[193] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[194] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[195] Zeyi Wen, Jiashuai Shi, Qinbin Li, Bingsheng He, and Jian Chen. ThunderSVM: A fast SVM library on GPUs and CPUs. *Journal of Machine Learning Research*, 19:797–801, 2018.

[196] Ting Chen, Xiaohua Zhai, Marvin Ritter, Mario Lucic, and Neil Houlsby. Self-supervised gans via auxiliary rotation loss. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12154–12163, 2019.

[197] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *CVPR*, pages 8188–8197, 2020.

[198] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[199] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, pages 2672–2680, 2014.

[200] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016.

[201] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *ECCV*, pages 649–666. Springer, 2016.

[202] Richard Zhang, Jun-Yan Zhu, Phillip Isola, Xinyang Geng, Angela S Lin, Tianhe Yu, and Alexei A Efros. Real-time user-guided image colorization with learned deep priors. *ACM TOG*, 36(4):119, 2017.

[203] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew P Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017.

[204] Jingwen Chen, Jiawei Chen, Hongyang Chao, and Ming Yang. Image blind denoising with generative adversarial network based noise modeling. In *CVPR*, pages 3155–3164, 2018.

[205] Ricardo Martin-Brualla, Rohit Pandey, Shuoran Yang, Pavel Pidlypenskyi, Jonathan Taylor, Julien Valentin, Sameh Khamis, Philip Davidson, Anastasia Tkach, Peter Lincoln, Adarsh Kowdle, Christoph Rhemann, Dan B Goldman, Cem Keskin, Steve Seitz, Shahram Izadi, and Sean Fanello. LookinGood: Enhancing performance capture with real-time neural re-rendering. In *Proc. SIGGRAPH Asia*, 2018.

[206] Moustafa Meshry, Dan B Goldman, Sameh Khamis, Hugues Hoppe, Rohit Pandey, Noah Snavely, and Ricardo Martin-Brualla. Neural rerendering in the wild. *CVPR*, 2019.

[207] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM TOG*, 2019.

[208] Hao Dong, Simiao Yu, Chao Wu, and Yike Guo. Semantic image synthesis via adversarial learning. In *ICCV*, 2017.

[209] Xiaolong Wang and Abhinav Gupta. Generative image modeling using style and structure adversarial networks. In *ECCV*, 2016.

[210] Zhifei Zhang, Yang Song, and Hairong Qi. Age progression/regression by conditional adversarial autoencoder. In *CVPR*, 2017.

[211] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *CVPR*, pages 8110–8119, 2020.

[212] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. *Advances in Neural Information Processing Systems*, 34:852–863, 2021.

[213] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *NeurIPS*, 2017.

[214] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *ECCV*, 2018.

[215] Hsin-Ying Lee, Hung-Yu Tseng, Qi Mao, Jia-Bin Huang, Yu-Ding Lu, Maneesh Singh, and Ming-Hsuan Yang. Drit++: Diverse image-to-image translation via disentangled representations. *IJCV*, pages 1–16, 2020.

[216] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016.

[217] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017.

[218] Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. Dualgan: Unsupervised dual learning for image-to-image translation. In *Proceedings of the IEEE international conference on computer vision*, pages 2849–2857, 2017.

[219] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. In *International conference on machine learning*, pages 1857–1865. PMLR, 2017.

[220] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *NeurIPS*, 2017.

[221] Minjun Li, Haozhi Huang, Lin Ma, Wei Liu, Tong Zhang, and Yugang Jiang. Unsupervised image-to-image translation with stacked cycle-consistent adversarial networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 184–199, 2018.

[222] Wayne Wu, Kaidi Cao, Cheng Li, Chen Qian, and Chen Change Loy. Transgaga: Geometry-aware unsupervised image-to-image translation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8012–8021, 2019.

[223] Sagie Benaim and Lior Wolf. One-sided unsupervised domain mapping. *Advances in neural information processing systems*, 30, 2017.

[224] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, Kun Zhang, and Dacheng Tao. Geometry-consistent generative adversarial networks for one-sided unsupervised domain mapping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2427–2436, 2019.

[225] 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.

[226] Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Kumar Singh, and Ming-Hsuan Yang. Diverse image-to-image translation via disentangled representations. In *ECCV*, 2018.

[227] Taesung Park, Jun-Yan Zhu, Oliver Wang, Jingwan Lu, Eli Shechtman, Alexei Efros, and Richard Zhang. Swapping autoencoder for deep image manipulation. *Advances in Neural Information Processing Systems*, 33:7198–7211, 2020.

[228] Yanwu Xu, Shaoan Xie, Wenhao Wu, Kun Zhang, Mingming Gong, and Kayhan Batmanghelich. Maximum spatial perturbation consistency for unpaired image-to-image translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18311–18320, 2022.

[229] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807, 2018.

[230] Xintao Wang, Ke Yu, Chao Dong, Xiaoou Tang, and Chen Change Loy. Deep network interpolation for continuous imagery effect transition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1692–1701, 2019.

[231] Peihao Zhu, Rameen Abdal, Yipeng Qin, and Peter Wonka. Sean: Image synthesis with semantic region-adaptive normalization. In *CVPR*, pages 5104–5113, 2020.

[232] Pan Zhang, Bo Zhang, Dong Chen, Lu Yuan, and Fang Wen. Cross-domain correspondence learning for exemplar-based image translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5143–5153, 2020.

[233] Xingran Zhou, Bo Zhang, Ting Zhang, Pan Zhang, Jianmin Bao, Dong Chen, Zhongfei Zhang, and Fang Wen. Cocosnet v2: Full-resolution correspondence learning for image translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11465–11475, 2021.

[234] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24, 2009.

[235] 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*, 2015.

[236] Moustafa Meshry, Yixuan Ren, Larry S Davis, and Abhinav Shrivastava. Step: Style-based encoder pre-training for multi-modal image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3712–3721, 2021.

[237] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, pages 1501–1510, 2017.

[238] Cheng-Han Lee, Ziwei Liu, Lingyun Wu, and Ping Luo. Maskgan: Towards diverse and interactive facial image manipulation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[239] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part V 14*, pages 597–613. Springer, 2016.

[240] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[241] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, pages 586–595, 2018.

[242] 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. *Advances in neural information processing systems*, 30, 2017.

[243] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *ICLR*, 2014.

[244] Katja Schwarz, Yiyi Liao, and Andreas Geiger. On the frequency bias of generative models. *Advances in Neural Information Processing Systems*, 34:18126–18136, 2021.

[245] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[246] Egor Zakharov, Aliaksandra Shysheya, Egor Burkov, and Victor Lempitsky. Few-shot adversarial learning of realistic neural talking head models. In *CVPR*, pages 9459–9468, 2019.

[247] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. First order motion model for image animation. In *NeurIPS*, December 2019.

[248] Egor Burkov, Igor Pasechnik, Artur Grigorev, and Victor Lempitsky. Neural head reenactment with latent pose descriptors. In *CVPR*, pages 13786–13795, 2020.

[249] Egor Zakharov, Aleksei Ivakhnenko, Aliaksandra Shysheya, and Victor Lempitsky. Fast bi-layer neural synthesis of one-shot realistic head avatars. In *ECCV*, August 2020.

[250] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, 2014.

[251] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, et al. Deep face recognition. In *BMVC*, volume 1, page 6, 2015.

[252] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *International conference on machine learning*, pages 3481–3490. PMLR, 2018.

[253] Olivia Wiles, A Koepke, and Andrew Zisserman. X2face: A network for controlling face generation using images, audio, and pose codes. In *ECCV*, pages 670–686, 2018.

[254] J. S. Chung, A. Nagrani, and A. Zisserman. VoxCeleb2: Deep Speaker Recognition. In *INTERSPEECH*, 2018.

[255] Adrian Bulat and Georgios Tzimiropoulos. How far are we from solving the 2d & 3d face alignment problem?(and a dataset of 230,000 3d facial landmarks). In *ICCV*, pages 1021–1030, 2017.

[256] Commentary: 'The 1-inch-tall barrier of subtitles': Bong Joon Ho rightly calls out Hollywood myopia, January 2020. URL https://www.chicagotribune.com/entertainment/movies/ct-ent-subtitles-parasite-0109-20200107-dchcnhgj7nhl5fp4qlcnqgv6uy-sto.html.

[257] Ginette Vincendeau. Hollywood Babel: The Multiple Language Version. *Screen*, 29(2): 24–39, 03 1988. ISSN 0036-9543. doi: 10.1093/screen/29.2.24. URL https://doi.org/10.1093/screen/29.2.24.

[258] Christoph Bregler, Michele Covell, and Malcolm Slaney. Video rewrite: Driving visual speech with audio. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 353–360, 1997.

[259] Borong Liang, Yan Pan, Zhizhi Guo, Hang Zhou, Zhibin Hong, Xiaoguang Han, Junyu Han, Jingtuo Liu, Errui Ding, and Jingdong Wang. Expressive talking head generation with granular audio-visual control. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3387–3396, 2022.

[260] Supasorn Suwajanakorn, Steven M Seitz, and Ira Kemelmacher-Shlizerman. Synthesizing obama: learning lip sync from audio. *ACM Transactions on Graphics (ToG)*, 36(4):1–13, 2017.

[261] Yang Zhou, Xintong Han, Eli Shechtman, Jose Echevarria, Evangelos Kalogerakis, and Dingzeyu Li. Makelttalk: speaker-aware talking-head animation. *ACM Transactions On Graphics (TOG)*, 39(6):1–15, 2020.

[262] Yasheng Sun, Hang Zhou, Kaisiyuan Wang, Qianyi Wu, Zhibin Hong, Jingtuo Liu, Errui Ding, Jingdong Wang, Ziwei Liu, and Koike Hideki. Masked lip-sync prediction by audio-visual contextual exploitation in transformers. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–9, 2022.

[263] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[264] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[265] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[266] Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–10, 2022.

[267] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8162–8171. PMLR, 18–24 Jul 2021. URL `https://proceedings.mlr.press/v139/nichol21a.html`.

[268] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

[269] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[270] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[271] Tengfei Wang, Ting Zhang, Bo Zhang, Hao Ouyang, Dong Chen, Qifeng Chen, and Fang Wen. Pretraining is all you need for image-to-image translation. In *arXiv*, 2022.

[272] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[273] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[274] Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A Efros. Everybody dance now. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5933–5942, 2019.

[275] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Nikolai Yakovenko, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. In *NeurIPS*, 2018.

[276] 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 I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf.

[277] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[278] Gaurav Shrivastava and Abhinav Shrivastava. Diverse video generation using a gaussian process trigger. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=Qm7R_SdqTpT.

[279] Lele Chen, Zhiheng Li, Ross K Maddox, Zhiyao Duan, and Chenliang Xu. Lip movements generation at a glance. In *Proceedings of the European conference on computer vision (ECCV)*, pages 520–535, 2018.

[280] J. S. Chung and A. Zisserman. Out of time: automated lip sync in the wild. In *Workshop on Multi-view Lip-reading, ACCV*, 2016.

[281] Adrian Bulat and Georgios Tzimiropoulos. How far are we from solving the 2d & 3d face alignment problem? (and a dataset of 230,000 3d facial landmarks). In *International Conference on Computer Vision*, 2017.

[282] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[283] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

[284] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2): 3, 2022.

[285] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022.

[286] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2(3): 5, 2022.

[287] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023.

[288] Yuheng Li, Haotian Liu, Qingyang Wu, Fangzhou Mu, Jianwei Yang, Jianfeng Gao, Chunyuan Li, and Yong Jae Lee. Gligen: Open-set grounded text-to-image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22511–22521, 2023.

[289] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[290] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22500–22510, 2023.

[291] Ruifei He, Shuyang Sun, Xin Yu, Chuhui Xue, Wenqing Zhang, Philip Torr, Song Bai, and Xiaojuan Qi. Is synthetic data from generative models ready for image recognition? *arXiv preprint arXiv:2210.07574*, 2022.

[292] Mert Bulent Sariyildiz, Karteek Alahari, Diane Larlus, and Yannis Kalantidis. Fake it till you make it: Learning transferable representations from synthetic imagenet clones. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[293] Yonglong Tian, Lijie Fan, Phillip Isola, Huiwen Chang, and Dilip Krishnan. Stablerep: Synthetic images from text-to-image models make strong visual representation learners. *arXiv preprint arXiv:2306.00984*, 2023.

[294] Hanqing Zhao, Dianmo Sheng, Jianmin Bao, Dongdong Chen, Dong Chen, Fang Wen, Lu Yuan, Ce Liu, Wenbo Zhou, Qi Chu, et al. X-paste: Revisiting scalable copy-paste for instance segmentation using clip and stablediffusion. 2023.

[295] Golnaz Ghiasi, Yin Cui, Aravind Srinivas, Rui Qian, Tsung-Yi Lin, Ekin D Cubuk, Quoc V Le, and Barret Zoph. Simple copy-paste is a strong data augmentation method for instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2918–2928, 2021.

[296] Yi Ke Yun and Weisi Lin. Selfreformer: Self-refined network with transformer for salient object detection. *arXiv preprint arXiv:2205.11283*, 2022.

[297] Xuebin Qin, Zichen Zhang, Chenyang Huang, Masood Dehghan, Osmar Zaiane, and Martin Jagersand. U2-net: Going deeper with nested u-structure for salient object detection. volume 106, page 107404, 2020.

[298] Yukun Su, Jingliang Deng, Ruizhou Sun, Guosheng Lin, and Qingyao Wu. A unified transformer framework for group-based segmentation: Co-segmentation, co-saliency detection and video salient object detection, 2022.

[299] Timo Lüddecke and Alexander Ecker. Image segmentation using text and image prompts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7086–7096, June 2022.

[300] Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5356–5364, 2019.

[301] Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–10, 2022.

[302] Andrey Voynov, Kfir Aberman, and Daniel Cohen-Or. Sketch-guided text-to-image diffusion models. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–11, 2023.

[303] Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18392–18402, 2023.

[304] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. *arXiv preprint arXiv:2108.01073*, 2021.

[305] Lukas Höllein, Ang Cao, Andrew Owens, Justin Johnson, and Matthias Nießner. Text2room: Extracting textured 3d meshes from 2d text-to-image models. *arXiv preprint arXiv:2303.11989*, 2023.

[306] Or Patashnik, Daniel Garibi, Idan Azuri, Hadar Averbuch-Elor, and Daniel Cohen-Or. Localizing object-level shape variations with text-to-image diffusion models. *arXiv preprint arXiv:2303.11306*, 2023.

[307] Kai Chen, Enze Xie, Zhe Chen, Lanqing Hong, Zhenguo Li, and Dit-Yan Yeung. Integrating geometric control into text-to-image diffusion models for high-quality detection data generation via text prompt. *arXiv: 2306.04607*, 2023.

[308] Zipeng Qi, Guoxi Huang, Zebin Huang, Qin Guo, Jinwen Chen, Junyu Han, Jian Wang, Gang Zhang, Lufei Liu, Errui Ding, et al. Layered rendering diffusion model for zero-shot guided image synthesis. *arXiv preprint arXiv:2311.18435*, 2023.

[309] Georgios Georgakis, Arsalan Mousavian, Alexander C Berg, and Jana Kosecka. Synthesizing training data for object detection in indoor scenes. *Robotics: Science and Systems (RSS)*, 2017.

[310] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, V. Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1082–10828, 2018. URL https://api.semanticscholar.org/CorpusID:4929980.

[311] Param S. Rajpura, Ravi Sadananda Hegde, and Hristo Bojinov. Object detection using deep cnns trained on synthetic images. *ArXiv*, abs/1706.06782, 2017. URL https://api.semanticscholar.org/CorpusID:2981790.

[312] Debidatta Dwibedi, Ishan Misra, and Martial Hebert. Cut, paste and learn: Surprisingly easy synthesis for instance detection. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1310–1319, 2017. URL https://api.semanticscholar.org/CorpusID:2229234.

[313] Haoshu Fang, Jianhua Sun, Runzhong Wang, Minghao Gou, Yong-Lu Li, and Cewu Lu. Instaboost: Boosting instance segmentation via probability map guided copy-pasting. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 682–691, 2019. URL https://api.semanticscholar.org/CorpusID:201126039.

[314] Nikita Dvornik, Julien Mairal, and Cordelia Schmid. Modeling visual context is key to augmenting object detection datasets. *ArXiv*, abs/1807.07428, 2018. URL https://api.semanticscholar.org/CorpusID:49882717.

[315] Ryan Burgert, Kanchana Ranasinghe, Xiang Li, and Michael S Ryoo. Peekaboo: Text to image diffusion models are zero-shot segmentors. *arXiv preprint arXiv:2211.13224*, 2022.

[316] Alexander C Li, Mihir Prabhudesai, Shivam Duggal, Ellis Brown, and Deepak Pathak. Your diffusion model is secretly a zero-shot classifier. *arXiv preprint arXiv:2303.16203*, 2023.

[317] Kevin Clark and Priyank Jaini. Text-to-image diffusion models are zero-shot classifiers. *arXiv preprint arXiv:2303.15233*, 2023.

[318] Daiqing Li, Huan Ling, Amlan Kar, David Acuna, Seung Wook Kim, Karsten Kreis, Antonio Torralba, and Sanja Fidler. Dreamteacher: Pretraining image backbones with deep generative models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16698–16708, 2023.

[319] Soumik Mukhopadhyay, Matthew Gwilliam, Vatsal Agarwal, Namitha Padmanabhan, Archana Swaminathan, Srinidhi Hegde, Tianyi Zhou, and Abhinav Shrivastava. Diffusion models beat gans on image classification. *arXiv preprint arXiv:2307.08702*, 2023.

[320] Luming Tang, Menglin Jia, Qianqian Wang, Cheng Perng Phoo, and Bharath Hariharan. Emergent correspondence from image diffusion. *arXiv preprint arXiv:2306.03881*, 2023.

[321] Grace Luo, Lisa Dunlap, Dong Huk Park, Aleksander Holynski, and Trevor Darrell. Diffusion hyperfeatures: Searching through time and space for semantic correspondence. In *Advances in Neural Information Processing Systems*, 2023.

[322] Weilai Xiang, Hongyu Yang, Di Huang, and Yunhong Wang. Denoising diffusion autoencoders are unified self-supervised learners. *arXiv preprint arXiv:2303.09769*, 2023.

[323] Shekoofeh Azizi, Simon Kornblith, Chitwan Saharia, Mohammad Norouzi, and David J Fleet. Synthetic data from diffusion models improves imagenet classification. *arXiv preprint arXiv:2304.08466*, 2023.

[324] Hritik Bansal and Aditya Grover. Leaving reality to imagination: Robust classification via generated datasets. *arXiv preprint arXiv:2302.02503*, 2023.

[325] Reyhane Askari Hemmat, Mohammad Pezeshki, Florian Bordes, Michal Drozdzal, and Adriana Romero-Soriano. Feedback-guided data synthesis for imbalanced classification. *arXiv preprint arXiv:2310.00158*, 2023.

[326] Shaobo Lin, Kun Wang, Xingyu Zeng, and Rui Zhao. Explore the power of synthetic data on few-shot object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 638–647, 2023.

[327] Haoyang Fang, Boran Han, Shuai Zhang, Su Zhou, Cuixiong Hu, and Wen-Ming Ye. Data augmentation for object detection via controllable diffusion models. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1257–1266, 2024.

[328] Yunhao Ge, Jiashu Xu, Brian Nlong Zhao, Neel Joshi, Laurent Itti, and Vibhav Vineet. Beyond generation: Harnessing text to image models for object detection and segmentation. *arXiv preprint arXiv:2309.05956*, 2023.

[329] Lihe Yang, Xiaogang Xu, Bingyi Kang, Yinghuan Shi, and Hengshuang Zhao. Freemask: synthetic images with dense annotations make stronger segmentation models. *Advances in Neural Information Processing Systems*, 36, 2024.

[330] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Probabilistic two-stage detection. *arXiv preprint arXiv:2103.07461*, 2021.

[331] Christophschuhmann. Christophschuhmann/improved-aesthetic-predictor: Clip+mlp aesthetic score predictor. URL `https://github.com/christophschuhmann/improved-aesthetic-predictor`.

[332] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask r-cnn. 2017. URL `https://api.semanticscholar.org/CorpusID:54465873`.

[333] Xiaofeng Mao, Yuefeng Chen, Yao Zhu, Da Chen, Hang Su, Rong Zhang, and Hui Xue. Coco-o: A benchmark for object detectors under natural distribution shifts. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6339–6350, 2023.

[334] 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. *ICLR*, 2021.

[335] Matthew Walmer, Saksham Suri, Kamal Gupta, and Abhinav Shrivastava. Teaching matters: Investigating the role of supervision in vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7486–7496, 2023.

[336] Sukmin Yun, Hankook Lee, Jaehyung Kim, and Jinwoo Shin. Patch-level representation learning for self-supervised vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8354–8363, June 2022.

[337] Adrian Ziegler and Yuki M Asano. Self-supervised learning of object parts for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14502–14511, 2022.

[338] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[339] Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. Exploring plain vision transformer backbones for object detection. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part IX*, pages 280–296. Springer, 2022.

[340] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Algorithms for learning kernels based on centered alignment. *ArXiv*, abs/1203.0550, 2012. URL `https://api.semanticscholar.org/CorpusID:9137763`.

[341] Xinlei Chen*, Saining Xie*, and Kaiming He. An empirical study of training self-supervised vision transformers. *arXiv preprint arXiv:2104.02057*, 2021.

[342] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021.

[343] Alaaeldin Ali, Hugo Touvron, Mathilde Caron, Piotr Bojanowski, Matthijs Douze, Armand Joulin, Ivan Laptev, Natalia Neverova, Gabriel Synnaeve, Jakob Verbeek, et al. Xcit: Cross-covariance image transformers. *Advances in neural information processing systems*, 34:20014–20027, 2021.

[344] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.

[345] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12179–12188, 2021.

[346] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7262–7272, 2021.

[347] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6836–6846, 2021.

[348] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *ICML*, volume 2, page 4, 2021.

[349] Yufei Xu, Jing Zhang, Qiming Zhang, and Dacheng Tao. Vitpose: Simple vision transformer baselines for human pose estimation. *Advances in Neural Information Processing Systems*, 35:38571–38584, 2022.

[350] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6824–6835, 2021.

[351] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 568–578, 2021.

[352] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22–31, 2021.

[353] Stéphane d'Ascoli, Hugo Touvron, Matthew L Leavitt, Ari S Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. In *International Conference on Machine Learning*, pages 2286–2296. PMLR, 2021.

[354] Shashank Shekhar, Florian Bordes, Pascal Vincent, and Ari Morcos. Objectives matter: Understanding the impact of self-supervised objectives on vision transformer representations. *arXiv preprint arXiv:2304.13089*, 2023.

[355] Weimin Tan, Bo Yan, and Bahetiyaer Bare. Feature super-resolution: Make machine see more clearly. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3994–4002, 2018.

[356] Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. Vision transformer adapter for dense predictions. In *The Eleventh International Conference on Learning Representations*, 2022.

[357] Hongyuan Chen, Yanting Pei, Hongwei Zhao, and Yaping Huang. Super-resolution guided knowledge distillation for low-resolution image classification. *Pattern Recognition Letters*, 155:62–68, 2022.

[358] Mingjian Zhu, Kai Han, Chao Zhang, Jinlong Lin, and Yunhe Wang. Low-resolution visual recognition via deep feature distillation. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3762–3766. IEEE, 2019.

[359] Assaf Shocher, Nadav Cohen, and Michal Irani. "zero-shot" super-resolution using deep internal learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3118–3126, 2018.

[360] Daniel Glasner, Shai Bagon, and Michal Irani. Super-resolution from a single image. In *2009 IEEE 12th international conference on computer vision*, pages 349–356. IEEE, 2009.

[361] Maria Zontak and Michal Irani. Internal statistics of a single natural image. In *CVPR 2011*, pages 977–984. IEEE, 2011.

[362] Yangtao Wang, Xi Shen, Yuan Yuan, Yuming Du, Maomao Li, Shell Xu Hu, James L Crowley, and Dominique Vaufreydaz. Tokencut: Segmenting objects in images and videos with self-supervised transformer and normalized cut. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[363] 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1201–1210, 2015.

[364] Thomas Deselaers, Bogdan Alexe, and Vittorio Ferrari. Localizing objects while learning their appearance. In *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV 11*, pages 452–466. Springer, 2010.

[365] 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.

[366] Sai Saketh Rambhatla, Rama Chellappa, and Abhinav Shrivastava. The pursuit of knowledge: Discovering and localizing novel categories using dual memory. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9153–9163, 2021.

[367] Van Huy Vo, Elena Sizikova, Cordelia Schmid, Patrick Pérez, and Jean Ponce. Large-scale unsupervised object discovery. *Advances in Neural Information Processing Systems*, 34: 16764–16778, 2021.

[368] Saksham Suri, Fanyi Xiao, Animesh Sinha, Sean Culatana, Raghuraman Krishnamoorthi, Chenchen Zhu, and Abhinav Shrivastava. Gen2det: Generate to detect. In *Synthetic Data for Computer Vision Workshop@ CVPR 2024*, 2024.

[369] Haotian Liu, Kilho Son, Jianwei Yang, Ce Liu, Jianfeng Gao, Yong Jae Lee, and Chunyuan Li. Learning customized visual models with retrieval-augmented knowledge. In

Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 15148–15158, 2023.

[370] Ahmet Iscen, Alireza Fathi, and Cordelia Schmid. Improving image recognition by retrieving from web-scale image-text data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19295–19304, 2023.

[371] Alexander Long, Wei Yin, Thalaiyasingam Ajanthan, Vu Nguyen, Pulak Purkait, Ravi Garg, Alan Blair, Chunhua Shen, and Anton van den Hengel. Retrieval augmented classification for long-tail visual recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6959–6969, 2022.

[372] Ahmet Iscen, Mathilde Caron, Alireza Fathi, and Cordelia Schmid. Retrieval-enhanced contrastive vision-text models. *arXiv preprint arXiv:2306.07196*, 2023.

[373] Zhuolin Yang, Wei Ping, Zihan Liu, Vijay Korthikanti, Weili Nie, De-An Huang, Linxi Fan, Zhiding Yu, Shiyi Lan, Bo Li, et al. Re-vilm: Retrieval-augmented visual language model for zero and few-shot image captioning. *arXiv preprint arXiv:2302.04858*, 2023.

[374] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023.

[375] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *Advances in neural information processing systems*, 32, 2019.

[376] Rohit Girdhar, Alaaeldin El-Nouby, Zhuang Liu, Mannat Singh, Kalyan Vasudev Alwala, Armand Joulin, and Ishan Misra. Imagebind: One embedding space to bind them all. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15180–15190, 2023.

[377] Yuanhuiyi Lyu, Xu Zheng, Jiazhou Zhou, and Lin Wang. Unibind: Llm-augmented unified and balanced representation space to bind them all. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26752–26762, 2024.

[378] Yuanhuiyi Lyu, Xu Zheng, Dahun Kim, and Lin Wang. Omnibind: Teach to build unequal-scale modality interaction for omni-bind of all. *arXiv preprint arXiv:2405.16108*, 2024.

[379] Saksham Suri, Saketh Rambhatla, Rama Chellappa, and Abhinav Shrivastava. Sparsedet: Improving sparsely annotated object detection with pseudo-positive mining. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6770–6781, 2023.

[380] Sharath Girish, Saksham Suri, Sai Saketh Rambhatla, and Abhinav Shrivastava. Towards discovery and attribution of open-world gan generated images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14094–14103, 2021.

[381] Saksham Suri, Moustafa Meshry, Larry S Davis, and Abhinav Shrivastava. Grit: Gan residuals for paired image-to-image translation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 4965–4975, 2024.

[382] Moustafa Meshry, Saksham Suri, Larry S Davis, and Abhinav Shrivastava. Learned spatial representations for few-shot talking-head synthesis. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 13829–13838, 2021.

[383] Soumik Mukhopadhyay, Saksham Suri, Ravi Teja Gadde, and Abhinav Shrivastava. Diff2lip: Audio conditioned diffusion models for lip-synchronization. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 5292–5302, 2024.

[384] Saksham Suri, Matthew Walmer, Kamal Gupta, and Abhinav Shrivastava. Lift: A surprisingly simple lightweight feature transform for dense vit descriptors. *arXiv preprint arXiv:2403.14625*, 2024.