ABSTRACT

Title of Dissertation: LEARNING AND COMPOSING PRIMITIVES FOR THE VISUAL WORLD

Kamal Gupta
Doctor of Philosophy, 2023

Dissertation Directed by: Professor Abhinav Shrivastava
Department of Computer Science

Professor Larry S. Davis
Department of Computer Science

Compositionality is at the core of how humans understand and recreate the visual world. It is what allows us to express infinitely many concepts using finite primitives. For example, we understand images as a combination of objects, videos as comprising of actions, or we generate 3D animations by rendering 3D surfaces with textures, materials, and lighting. It is unsurprising to see composition also appear in almost all human-created art forms such as language, music, design, or even mathematics. Although compositionality seems an obvious and prevalent way humans consume and create data, it is often eluded in computational approaches such as deep learning. Current systems often assume the availability of exhaustive labeled concepts or primitives during training and fail to generalize to new compositions during inference. In this dissertation, we propose to discover compositional primitives from the data with little to no supervision and show how we can use these primitives for improving generalization in real-world

applications such as classification, correspondence, or 2D/3D synthesis.

In the first half of this dissertation, I propose two complementary approaches to discover compositional discrete primitives from visual data. Given a large collection of images without labels, I propose a generative and a contrastive way of recognizing discriminative parts in the image which are usual for visual recognition. In the generative approach, I take inspiration from bayesian approaches such as variational autoencoders, to develop a system that can express images in form of discrete language-like representation. In the contrastive approach, I play a referential game between two neural network agents, to learn meaningful discrete concepts from images. I further show applications of these approaches in image and video editing by learning a dense correspondence of primitives across images.

In the second half, I'll focus on learning how to compose primitives for both 2D and 3D visual data. By expressing the scenes as an assembly of smaller parts, we can easily perform generation from scratch or from partial scenes as input. I present two works, one on composing multiple viewpoints to synthesize 3D objects, and another work on composing bounding boxes or cuboids to generate scene layouts. I also review a work on discovering a data-driven way of ordering traversing an image or a scene, for composition. I show applications of these works in image/video compression, as well as 2D and 3D content creation.

LEARNING AND COMPOSING PRIMITIVES FOR THE VISUAL WORLD

by

Kamal Gupta

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
2023

Advisory Committee:
Professor Abhinav Shrivastava, Chair/Advisor
Professor Larry S. Davis, Co-chair/Advisor
Professor Carol Y. Espy-Wilson, Dean's Representative
Professor Matthias Zwicker
Professor Noah Snavely, Cornell University

To my family and friends,

for your unwavering and unconditional support, and I am eternally grateful.

Acknowledgments

This thesis is a culmination of a joint effort of many individuals and institutions over the course of the last 5 years and I want to express my deep appreciation and gratitude to all those who have contributed to its successful completion.

First and foremost, I would like to express my sincere gratitude to my wife, Gowthami Somepalli, for pushing me to apply for a Ph.D. after I spent years procrastinating. Thanks for sticking with me through the thick and thin, endless brainstorming, bearing my rants, and helping me prepare for the interviews and presentations (in no specific order). I want to thank my parents for always pushing me to be a better person, my sisters Ruchi, Shilpa, and Sravani, my niece Sanvi and nephew Ibhan, and my brother-in-laws Himanshu and Ankur, for their unconditional love and support. Their patience, understanding, and love provided me with the motivation to overcome the many challenges I faced while completing this thesis.

I want to thank my advisor Abhinav Shrivastava for his unwavering support, guidance, and constructive criticism in making this thesis happen. I especially appreciate his flexibility and availability every time I needed it, the freedom to explore various research topics that I was passionate about, and above all, his friendship throughout the last 5 years. I am privileged to have his expertise, knowledge, and enthusiasm which were essential to the successful completion of my dissertation. I also want to thank my co-advisor, Larry Davis, for being a great mentor, teaching me how to critically evaluate my own ideas, and creating a massive Computer Vision

community at the University of Maryland that I immensely benefited from.

I am also grateful to other professors and mentors I collaborated with and learned from - especially Matthias Zwicker for his valuable insights, and for introducing me to the world of graphics, David Jacobs for a number of discussions, Noah Snavely, Ameesh Makadia, and Vijay Mahadevan for their expertise and constructive criticism which were instrumental in shaping the direction of my research. I thank David Luebke and Stephano Soatto for building the best teams that I had the pleasure to be part of, Sanjiv Singh, Stephen Nuske, Subhajit Sanyal and Vidit Jain, for their mentorship before I even started my Ph.D. I sincerely appreciate the time and effort they dedicated to teaching me and reviewing my work.

I want to thank DARPA SAIL-ON (W911NF2020009), DARPA SemaFor (HR001119S0085), and DARPA GARD (HR00112020007) programs, Amazon Research Award to Abhinav, Kulkarni Fellowship, Dean's Fellowship, Graduate Assistant Award, and Google, NVIDIA, and Amazon Research Labs which helped me conduct my research.

I want to express my gratitude to my friends/collaborators (Saurabh, Justin, Alessandro, Susmija, Ketul, Hanyu, Abhishek, Varun, Carlos, Nicholas, Towaki, Sameh, Anubhav, Sharath, Hao, Shishira, Matt W., Saksham, Nirat, Vinoj, Khoi, Archana), friends/labmates (Pulkit, Luyu, Pallabi, Lillian, Saketh, Gaurav, Soumik, Neha, Sahil, Moustafa, Max, Bo, Matt G., Shirley, Alex, Mara, Yixuan, Shlok, Anshul, Aman, Pranav, Sweta, Namitha, Vatsal, Ahmed, Ameya, Samyadeep, Sneha, Rohan, Trisha, Uttaran, Philip, Amanpreet, Koutilya, Sharmila, Kevin, Shramay), and friends/housemates (Bhavesh and Shivani) for their unwavering support and encouragement throughout this journey.

I would like to express my gratitude to Tom Hurst, and other staff and faculty of the Computer Science Department at UMD and UMIACS who were super supportive and responsive

when it came to any bureaucratic or compute-related questions.

Last but not the least, I want to thank the essential workers and healthcare professionals across the world, who put their lives at risk every day, to help us make it through the Covid-19 pandemic. Like many others in my time, I owe them a significant deal, and while I can't name all of them, my Ph.D. wouldn't have been possible without their tireless efforts.

My sincere apologies to those I've inadvertently left out. 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 (and even animals) [10] can innately imagine and recognize a large number of objects in diverse scenes, by composing a few known objects and their attributes. However, this ability to synthesize and recognize infinitely many new combinations from finite concepts called *compositional generalization*, remains an Achilles's heel of current data-driven approaches (*e.g.*, deep networks) that assume robust labeled training data is available for exhaustive object and their properties [11].

In order to harness the computational approaches for building real-world applications, it is crucial for them to understand and perform composition. George Bool, also known as the father of modern boolean algebra, formally defined the *principle of compositionality* as following,

"The meaning of a complex expression is determined by the meanings of its constituent expressions and the rules used to combine them"

The majority of this thesis involves abstracting and solving these two challenges in compositional learning:

1. Discovering discrete primitives from data (images and videos) with few or no labels which can be composed for various downstream applications such as segmentation, correspon-

dence, or image retrieval.

2. Learning the rules to combine visual primitives meaningfully to generate rich, and diverse compositions of the input data (images and 3D).

Being able to understand and compose 2D and 3D content (both static and dynamic) will unlock the next generation of tools empowered by Machine Learning. It will allow artificial agents to understand the visual world around them and perform repetitive or dangerous tasks reducing human time and effort by orders of magnitude. Tools to create content will enable the creation of large labeled highly realistic synthetic datasets. They will also be quintessential for entertainment (movies, sports, gaming) companies as well as have applications in robotics, healthcare, education, and e-commerce.

1.2 Discovering Primitives from Data

In order to better understand the guiding principles behind generative process of data, it is imperative to build representations that can discover simple parts or primitives that compose the data. In images, representations learned by current popular unsupervised and self-supervised machine learning systems, however, follow a distributed paradigm. This means that image is usually represented using a high-dimensional continuous vector where a single dimension can contribute to multiple concepts, and multiple dimensions can contribute to a single concept. In this work, we propose a new paradigm to represent images in the form of discrete primitives that appear in images, and can be composed to either reconstruct an image or to differentiate an image from other images.

We discuss two foundational works, PatchVAE in Chapter 2 and PatchGame in Chap-

Figure 1.1: We propose to learns mid-level patches in images which occur repetitively across the data. We further show that these mid-level patches can cluster meaningfully and can also enable dense correspondence applications.

ter 3. PatchVAE [12] attempts to model the images by representing them as a combination of part appearance maps, and part visibility maps, which combined together allows us to reconstruct original images. These parts are meaningful across a collection of images and allow image editing applications by simply replacing discrete symbols from an image by another image. PatchGame [13] follows a constrastive approach instead. In this work, we play a cooperative referential game between two neural network agents, a speaker and a listener, who can communicate with each other only via a discrete bottleneck. The goal for the listener is to match a discrete message sent by the speaker to the corresponding image. I show the applications of this discrete bottleneck in downstream applications such as detection, retrieval, and part segmentations. Fur-

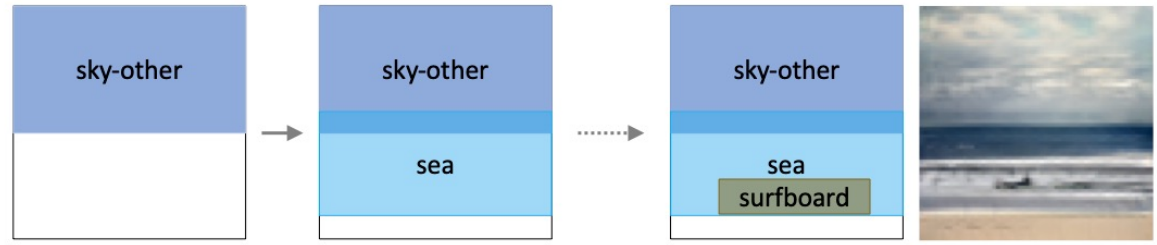
ther in Chapter 4, ASIC [14] takes these discrete primitives a step further by learning a dense correspondence for a set of images as shown in Fig. 1.1.

Going beyond images, the principle of compositionality can also be applied to represent and train fast, efficient neural networks. In LilNetX [15, 16], we learn a set of discrete weights sampled from a codebook, to provide a replacement for weights of standard CNNs such as ResNets or MobileNets. These discrete weights combined with entropy regularization, allow for highly quantized, and sparse neural networks that can be used for tasks such as classification, or detection. An extension of this work (under review) also shows the application of our entropy regularization technique to compress Neural Fields. While in this dissertation, we restrict ourselves to composition for the visual data, we encourage readers to check the above-mentioned works on model compression.

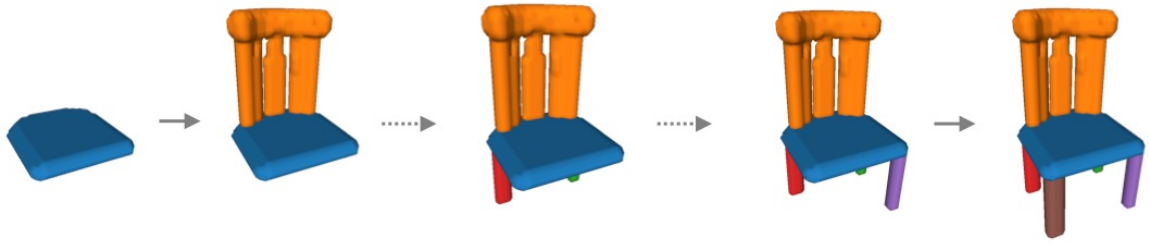
1.3 Learning to Compose Primitives

The second part of this dissertation focus on the process of synthesizing or learning a meaningful arrangement of some known concepts to give rise to novel scenes with desired characteristics.

In Chapter 5, our primitives are depth maps from different camera viewpoints. By synthesizing and composing these multiview depthmaps, we propose a new way of generating 3D surfaces. In Chapter 6, we consider primitives to be bounding boxes, or cuboids for 2D and 3D scenes. This arrangement of concepts or primitives in a scene, can be referred to as layout of the scene and it can be characterized by contextual relationships (e.g., support, occlusion, and relative likelihood, position, and size) between objects comprising the scene. In this work, also



(a) Autoregressive 2D layout generation and downstream Layout-to-Image application



(b) Generating 3D objects autoregressively

Figure 1.2: LayoutTransformer [7] can synthesize layouts in diverse natural as well as human designed data domains such as documents, mobile app wireframes, natural scenes or 3D objects in a sequential manner.

dubbed, LayoutTransformer [7], we propose an approach to autoregressively generate 2D and 3D layouts from scratch or from an initial seed set of primitives, and can easily scale to support an arbitrary of primitives per layout. A shortcoming of this approach is that it can only generate layouts elements in a raster scan order. We address this shortcoming in Neural SFC [17] discussed in Chapter 7, where we learn a data driven way to learn scan order of images.

Part I: Discovering Primitives from Data

Chapter 2: Learning Local Latent Codes for Recognition

Due to the availability of large labeled visual datasets, supervised learning has become the dominant paradigm for visual recognition. That is, to learn about any new concept, the modus operandi is to collect thousands of labeled examples for that concept and train a powerful classifier, such as a deep neural network. This is necessary because the current generation of models based on deep neural networks require large amounts of labeled data [18]. This is in stark contrast to the insights that we have from developmental psychology on how infants develop perception and cognition without any explicit supervision [19]. Moreover, the supervised learning paradigm is ill-suited for applications, such as health care and robotics, where annotated data is hard to obtain either due to privacy concerns or high cost of expert human annotators. In such cases, learning from very few labeled images or discovering underlying natural patterns in large amounts of unlabeled data can have a large number of potential applications. Discovering such patterns from unlabeled data is the standard setup of unsupervised learning.

Over the past few years, the field of unsupervised learning in computer vision has followed two seemingly different tracks with different goals: generative modeling and self-supervised learning. The goal of generative modeling is to learn the probability distribution from which data was generated, given some training data. Such models, learned using reconstruction-based losses, can draw samples from the same distribution or evaluate the likelihoods of new data,

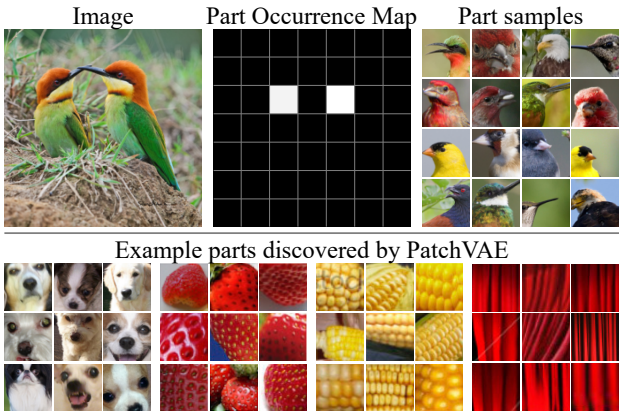


Figure 2.1: PatchVAE learns to encode repetitive parts across a dataset, by modeling their appearance and occurrence. (top) Given an image, the occurrence map of a particular part learned by PatchVAE is shown in the middle, capturing the head/beak of the birds. Samples of the same part from other images are shown on the right, indicating consistent appearance. (bottom) More examples of parts discovered by our PatchVAE framework.

and are useful for learning compact representation of images. However, we argue that these representations are not as useful for visual recognition. This is not surprising since the task of reconstructing images does not require the bottleneck representation to sort out meaningful data useful for recognition and discard the rest; on the contrary, it encourages preserving as much information as possible for reconstruction.

In comparison, the goal in self-supervised learning is to learn representations that are useful for recognition. The standard paradigm is to establish proxy tasks that don't require human-supervision but can provide signals useful for recognition. Due to the mismatch in goals of unsupervised learning for visual recognition and the representations learned from generative modeling, self-supervised learning is a more popular way of learning representations from unlabeled data. However, fundamental limitation of this self-supervised paradigm is that we need to define a proxy-task that can mimic the desired recognition task. It is not possible to always establish such a task, nor are these tasks generalizable across recognition tasks.

In this paper, our goal is to enable the unsupervised generative modeling approach of VAEs

to learn representations useful for recognition. Our key hypothesis is that for a representation to be useful, it should capture just the *interesting* parts of the images, as opposed to *everything* in the images.

What constitutes an interesting image part has been defined and studied in earlier works that pre-date the end-to-end trained deep network methods [20, 21, 22]. Taking inspiration from these works, we propose a novel representation that only encodes few such parts of an image that are repetitive across the dataset, i.e., the patches that occur often in images. By avoiding reconstruction of the entire image our method can focus on regions that are repeating and consistent across many images. In an encoder-decoder based generative model, we constrain the encoder architecture to learn such repetitive parts – both in terms of representations for appearance of these parts (or patches in an image) and where these parts occur. We formulate this using variational auto-encoder (β-VAEs) [23, 24], where we impose novel structure on the latent representations. We use discrete latents to model part presence or absence and continuous latents to model their appearance. Figure 2.1 shows an example of the discrete latents or occurrence map, and example parts discovered by our approach, PatchVAE. We present PatchVAE in Section 2.2 and demonstrate that it learns representations that are much better for recognition as compared to those learned by the standard β-VAEs [23, 24].

In addition, we present losses that favor foreground, which is more likely to contain repetitive patterns, in Section 2.2.4, and demonstrate that they result in representations that are much better at recognition. Finally, in Section 2.3, we present results on CIFAR100 [25], MIT Indoor Scene Recognition [26], Places [27], and ImageNet [28] datasets. To summarize, our contributions are as follows:

- We propose a novel patch-based bottleneck in the VAE framework that learns representations that can encode repetitive parts across images.
- We demonstrate that our method, PatchVAE, learns unsupervised representations that are better suited for recognition in comparison to traditional VAEs.
- We show that losses that favor foreground are better for unsupervised representation learning for recognition.
- We perform extensive ablation analysis of the proposed PatchVAE architecture.

2.1 Related Work

Due to its potential impact, unsupervised learning (particularly for deep networks) is one of the most researched topics in visual recognition over the past few years. Generative models such as VAEs [23, 24, 29, 30], PixelRNN [31], PixelCNN [32, 33], and their variants have proven effective when it comes to learning compressed representation of images while being able to faithfully reconstruct them as well as draw samples from the data distribution. GANs [34, 35, 36, 37] on the other hand, while don't model the probability density explicitly, can still produce high quality image samples from noise. There has been work combining VAEs and GANs to be able to simultaneously learn image data distribution while being able to generate high quality samples from it [38, 39, 40]. Convolution sparse coding [41] is an alternative approach for reconstruction or image in-painting problems. Our work complements existing generative frameworks in that we provide a structured approach for VAEs that can learn beyond low-level representations. We show the effectiveness of the representations learned by our model by using them for visual recognition tasks.

There has been a lot of work in interpreting or disentangling representations learned using generative models such as VAEs [24, 42, 43]. However, there is little evidence of effectiveness of disentangled representations in visual recognition. Semi-supervised learning using generative models [44, 45], where partial or noisy labels are available to the model during training, has shown lots of promise in applications of generating conditioned samples from the model. In our work however, we focus on incorporating inductive biases in these generative models (e.g., VAEs) so they can learn representations better suited for visual recognition.

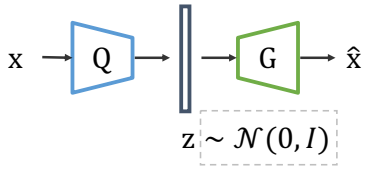
A related, but orthogonal, line of work is self-supervised learning where a proxy task is designed to learn representation useful for recognition. These proxy tasks vary from simple tasks like arranging patches in an image in the correct spatial order [46, 47] and arranging frames from a video in correct temporal order [48, 49], to more involved tasks like in-painting [50] and context prediction [51, 52]. We follow the best practices from this line of work for evaluating the learned representations.

2.2 Our Approach

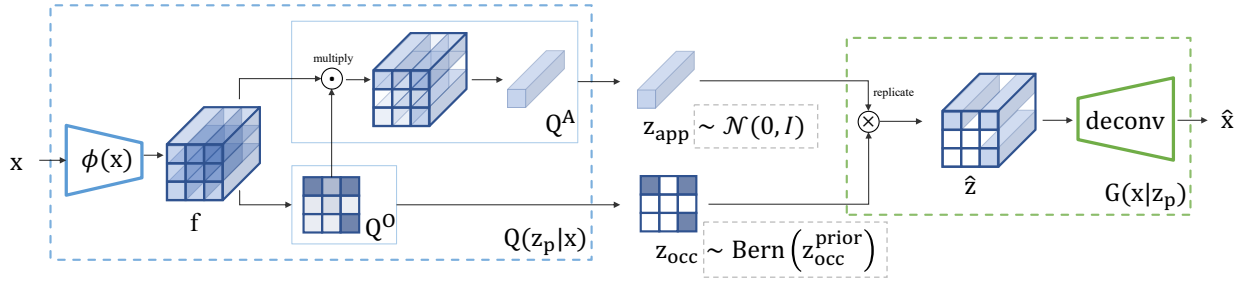
Our work builds upon VAE framework proposed by Kingma and Welling [23]. We briefly review relevant aspects of the VAE framework and then present our approach.

2.2.1 VAE Review

Standard VAE framework assumes a generative model for data where first a latent $\mathbf{z}$ is sampled from a prior $p(\mathbf{z})$ and then the data is generated from a conditional distribution $G(\mathbf{x}|\mathbf{z})$. A variational approximation $Q(\mathbf{z}|\mathbf{x})$ to the true intractable posterior is introduced and the model



(a) VAE Architecture



(b) PatchVAE Architecture

Figure 2.2: (a) **VAE Architecture**: In a standard VAE architecture, output of encoder network is used to parameterize the variational posterior for z. Samples from this posterior are input to the decoder network. (b) **Proposed PatchVAE Architecture**: Our encoder network computes a set of feature maps $\mathbf{f}$ using $\phi(\mathbf{x})$. This is followed by two independent single layer networks. The bottom network generates part occurrence parameters Q^{O}. We combine Q^{O} with output of top network to generate part appearance parameters Q^{A}. We sample $\mathbf{z}_{\mathrm{occ}}$ and $\mathbf{z}_{\mathrm{app}}$ to construct $\hat{\mathbf{z}}$ as described in Section 2.2.2 which is input to the decoder network. We also visualize the corresponding priors for latents $\mathbf{z}_{\mathrm{app}}$ and $\mathbf{z}_{\mathrm{occ}}$ in the dashed gray boxes.

is learned by minimizing the following negative variational lower bound (ELBO),

$$\begin{aligned}\mathcal{L}_{\text{VAE}}(\mathbf{x}) = &- \mathbb{E}_{z\sim Q(\mathbf{z}|\mathbf{x})}\left[\log G(\mathbf{x}|\mathbf{z})\right] \\ &+ D_{\text{KL}}\left[Q(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z})\right]\end{aligned} \tag{2.1}$$

where $Q(\mathbf{z}|\mathbf{x})$ is often referred to as an encoder as it can be viewed as mapping data to the the latent space, while $G(\mathbf{x}|\mathbf{z})$ is referred to as a decoder (or generator) that can be viewed as mapping latents to the data space. Both Q and G are commonly parameterized as neural networks. Figure 2.2a shows the commonly used VAE architecture. If the conditional $G(\mathbf{x}|\mathbf{z})$ takes a gaussian form, negative log likelihood in the first term of RHS of Equation (2.1) becomes mean squared error between generator output $\hat{\mathbf{x}} = G(\mathbf{x}|\mathbf{z})$ and input data $\mathbf{x}$. In the second term, prior $p(\mathbf{z})$ is assumed to be a multi-variate normal distribution with zero-mean and identity covariance $\mathcal{N}(0, \mathcal{I})$ and the loss simplifies to

$$\mathcal{L}_{\text{VAE}}(\mathbf{x}) = \|\mathbf{x} - \hat{\mathbf{x}}\|^2 + D_{\text{KL}}\left[Q(\mathbf{z}|\mathbf{x}) \parallel \mathcal{N}(0, \mathcal{I})\right] \tag{2.2}$$

When G and Q are differentiable, entire model can be trained with SGD using reparameterization trick [23]. Matthey et al. [24] propose an extension for learning disentangled representation by incorporating a weight factor β for the KL Divergence term yielding

$$\mathcal{L}_{\beta\text{VAE}}(\mathbf{x}) = \|\mathbf{x} - \hat{\mathbf{x}}\|^2 + \beta D_{\text{KL}}\left[Q(\mathbf{z}|\mathbf{x}) \parallel \mathcal{N}(0, \mathcal{I})\right] \tag{2.3}$$

VAE framework aims to learn a generative model for the images where the latents $\mathbf{z}$ repre-

sent the corresponding low dimensional generating factors. The latents $\mathbf{z}$ can therefore be treated as image representations that capture the necessary details about images. However, we postulate that representations produced by the standard VAE framework are not ideal for recognition as they are learned to capture *all* details, rather than capturing 'interesting' aspects of the data and dropping the rest. This is not surprising since there formulation does not encourage learning semantic information. For learning semantic representations, in the absence of any relevant supervision (as is available in self-supervised approaches), inductive biases have to be introduced. Therefore, taking inspiration from works on unsupervised mid-level pattern discovery [20, 21, 22], we propose a formulation that encourages the encoder to only encode such few parts of an image that are repetitive across the dataset, i.e., the patches that occur often in images.

Since the VAE framework provides a principled way of learning a mapping from image to latent space, we consider it ideal for our proposed extension. We chose β-VAEs for their simplicity and widespread use. In Section 2.2.2, we describe our approach in detail and in Section 2.2.4 propose a modification in the reconstruction error computation to bias the error term towards foreground high-energy regions (similar to the biased initial sampling of patterns in Singh et al. [20]).

2.2.2 PatchVAE

Given an image $\mathbf{x}$, let $\mathbf{f} = \phi(\mathbf{x})$ be a deterministic mapping that produces a 3D representation $\mathbf{f}$ of size $h \times w \times d_e$, with a total of $L = h \times w$ locations (grid-cells). We aim to encourage the encoder network to only encode parts of an image that correspond to highly repetitive patches. For example, a random patch of noise is unlikely to occur frequently, whereas patterns like faces,

wheels, windows, etc. repeat across multiple images. In order capture this intuition, we force the representation $\mathbf{f}$ to be useful for predicting frequently occurring parts in an image, and use *just* these predicted parts to reconstruct the image. We achieve this by transforming $\mathbf{f}$ to $\hat{\mathbf{z}}$ which encodes a set of parts at a small subset of L locations on the grid cells. We refer to $\hat{\mathbf{z}}$ as "patch latent codes" for an image. Next we describe how we re-tool the β-VAE framework to learn these local latent codes. We first describe our setup for a single part and follow it up with a generalization to multiple parts (Section 2.2.3).

Image Encoding. Given the image representation $\mathbf{f} = \phi(x)$, we want to learn part representations at each grid location l (where $l \in \{1, \ldots, L\}$). A part is parameterized by its appearance $\mathbf{z}_{\text{app}}$ and its occurrence $\mathbf{z}^l_{\text{occ}}$ (i.e., presence or absence of the part at grid location l). We use two networks, $Q^{\text{A}}_{\mathbf{f}}$ and $Q^{\text{O}}_{\mathbf{f}}$, to parameterize posterior distributions $Q^{\text{A}}_{\mathbf{f}}(\mathbf{z}_{\text{app}} \mid \mathbf{f})$ and $Q^{\text{O}}_{\mathbf{f}}(\mathbf{z}^l_{\text{occ}} \mid \mathbf{f})$ of the part parameters $\mathbf{z}_{\text{app}}$ and $\mathbf{z}^l_{\text{occ}}$ respectively. Since the mapping $\mathbf{f} = \phi(\mathbf{x})$ is deterministic, we can re-write these distributions as $Q^{\text{A}}_{\mathbf{f}}(\mathbf{z}_{\text{app}} \mid \phi(\mathbf{x}))$ and $Q^{\text{O}}_{\mathbf{f}}(\mathbf{z}^l_{\text{occ}} \mid \phi(\mathbf{x}))$; or simply $Q^{\text{A}}(\mathbf{z}_{\text{app}} \mid \mathbf{x})$ and $Q^{\text{O}}(\mathbf{z}^l_{\text{occ}} \mid \mathbf{x})$. Therefore, given an image $\mathbf{x}$ the encoder networks estimate the posterior $Q^{\text{A}}(\mathbf{z}_{\text{app}} \mid \mathbf{x})$ and $Q^{\text{O}}(\mathbf{z}^l_{\text{occ}} \mid \mathbf{x})$. Note that $\mathbf{f}$ is a deterministic feature map, whereas $\mathbf{z}_{\text{app}}$ and $\mathbf{z}^l_{\text{occ}}$ are stochastic.

Image Decoding. We utilize a generator or decoder network G, that given $\mathbf{z}_{\text{occ}}$ and $\mathbf{z}_{\text{app}}$, reconstructs the image. First, we sample a part appearance $\hat{\mathbf{z}}_{\text{app}}$ (d_p dimensional, continuous) and then sample part occurrence $\hat{\mathbf{z}}^l_{\text{occ}}$ (L dimensional, binary) one for each location l from the posteriors

$$\begin{aligned} \hat{\mathbf{z}}_{\text{app}} &\sim Q^{\text{A}}(\mathbf{z}_{\text{app}} \,|\, \mathbf{x}) \\ \hat{\mathbf{z}}^l_{\text{occ}} &\sim Q^{\text{O}}\left(\mathbf{z}^l_{\text{occ}} \,|\, \mathbf{x}\right), \quad \text{where } l \in \{1, \ldots, L\} \end{aligned} \tag{2.4}$$

Next, we construct a 3D representation $\hat{\mathbf{z}}$ by placing $\hat{\mathbf{z}}_{\text{app}}$ at every location l where the part is present (i.e., $\hat{\mathbf{z}}^l_{\text{occ}} = 1$). This can be implemented by a broadcasted product of $\hat{\mathbf{z}}_{\text{app}}$ and $\hat{\mathbf{z}}^l_{\text{occ}}$. We refer to $\hat{\mathbf{z}}$ as **patch latent code**. Again note that $\mathbf{f}$ is deterministic and $\hat{\mathbf{z}}$ is stochastic. Finally, a deconvolutional network takes $\hat{\mathbf{z}}$ as input and generates an image $\hat{\mathbf{x}}$. This image generation process can be written as

$$\hat{\mathbf{x}} \sim G\left(\mathbf{x} \mid \mathbf{z}^1_{\text{occ}}, \mathbf{z}^2_{\text{occ}}, \ldots, \mathbf{z}^L_{\text{occ}}, \mathbf{z}_{\text{app}}\right) \tag{2.5}$$

Since all latent variables ($\mathbf{z}^l_{\text{occ}}$ for all l and $\mathbf{z}_{\text{app}}$) are independent of each other, they can be stacked as

$$\mathbf{z}_{\text{p}} = \left[\mathbf{z}^1_{\text{occ}}; \mathbf{z}^2_{\text{occ}}; \ldots; \mathbf{z}^L_{\text{occ}}; \mathbf{z}_{\text{app}}\right]. \tag{2.6}$$

This enables us to use a simplified the notation (refer to Equation (2.4) and Equation (2.5)):

$$\begin{aligned} \hat{\mathbf{z}}_{\text{p}} &\sim Q^{\{\text{A,O}\}}(\mathbf{z}_{\text{p}} \mid \mathbf{x}) \\ \hat{\mathbf{x}} &\sim G\left(\mathbf{x} \mid \mathbf{z}_{\text{p}}\right) \end{aligned} \tag{2.7}$$

Note that despite the additional structure, our model still resembles the setup of variational auto-encoders. The primary difference arises from: (1) use of discrete latents for part occurrence, (2) patch-based bottleneck imposing additional structure on latents, and (4) feature assembly for generator.

Training. We use the training setup of β-VAE and use the maximization of variational lower bound to train the encoder and decoder jointly (described in Section 2.2.1). The posterior Q^{A},

which captures the appearance of a part, is assumed to be a Normal distribution with zero-mean and identity covariance $\mathcal{N}(0, \mathcal{I})$. The posterior Q^{O}, which captures the presence or absence a part, is assumed to be a Bernoulli distribution $\mathrm{Bern}\left(\mathbf{z}_{\mathrm{occ}}^{\mathrm{prior}}\right)$ with prior $\mathbf{z}_{\mathrm{occ}}^{\mathrm{prior}}$. Therefore, the ELBO for our approach can written as (refer to Equation (2.3)):

$$\begin{aligned}\mathcal{L}_{\text{PatchVAE}}(\mathbf{x}) = &- \mathbb{E}_{\mathbf{z}_{\mathrm{p}} \sim Q^{\{\mathrm{A,O}\}}(\mathbf{z}_{\mathrm{p}} \mid \mathbf{x})}\left[G\left(\mathbf{x} \mid \mathbf{z}_{\mathrm{p}}\right)\right] \\ &+ \beta D_{\mathrm{KL}}\left[Q^{\{\mathrm{A,O}\}}(\mathbf{z}_{\mathrm{p}} \mid \mathbf{x}) \parallel p(\mathbf{z}_{\mathrm{p}})\right]\end{aligned} \tag{2.8}$$

where, the D_{KL} term can be expanded as:

$$\begin{aligned}&D_{\mathrm{KL}}\left[Q^{\{\mathrm{A,O}\}}(\mathbf{z}_{\mathrm{p}} \mid \mathbf{x}) \parallel p(\mathbf{z}_{\mathrm{p}})\right] = \\ &\beta_{\text{app}} \sum_{l=1}^{L} D_{\mathrm{KL}}\left(Q^{\mathrm{O}}(\mathbf{z}_{\mathrm{occ}}^{l} \mid \mathbf{x}) \parallel \mathrm{Bern}\left(\mathbf{z}_{\mathrm{occ}}^{\mathrm{prior}}\right)\right) \\ &+ \beta_{\text{occ}} D_{\mathrm{KL}}\left(Q^{\mathrm{A}}(\mathbf{z}_{\text{app}} \mid \mathbf{x}) \parallel \mathcal{N}(0, \mathcal{I})\right)\end{aligned} \tag{2.9}$$

Implementation details. As discussed in Section 2.2.1, the first and second terms of the RHS of Equation (2.8) can be trained using L2 reconstruction loss and reparameterization trick [23]. In addition, we also need to compute KL Divergence loss for part occurrence. Learning discrete probability distribution is a challenging task since there is no gradient defined to backpropagate reconstruction loss through the stochastic layer at decoder even when using the reparameterization trick. Therefore, we use the relaxed-bernoulli approximation [53, 54] for training part occurrence distributions $\mathbf{z}_{\mathrm{occ}}^{l}$.

For an $H \times W$ image, network $Q(\mathbf{f} \mid \mathbf{x})$ first generates feature maps of size $(h \times w \times d_e)$, where (h, w) are spatial dimensions and d_e is the number of channels. Therefore, the number of locations $L = h \times w$. Encoders $Q_{\mathbf{f}}^{\mathrm{A}}(\mathbf{z}_{\text{app}} \mid \mathbf{f})$ and $Q_{\mathbf{f}}^{\mathrm{O}}(\mathbf{z}_{\mathrm{occ}}^{l} \mid \mathbf{f})$ are single layer neural networks to

compute $\mathbf{z}_{\text{app}}$ and $\mathbf{z}^l_{\text{occ}}$. $\mathbf{z}^l_{\text{occ}}$ is $(h \times w \times 1)$-dimensional multivariate bernoulli parameter and $\mathbf{z}_{\text{app}}$ is $(1 \times 1 \times d_p)$-dimensional multivariate gaussian. d_p is length of the latent vector for a single part. Input to the decoder $\hat{\mathbf{z}}$ is $(h \times w \times d_p)$-dimensional. In all experiments, we fix $h = \frac{H}{8}$ and $w = \frac{W}{8}$.

Constructing $\mathbf{z}_{\text{app}}$. Notice that $\mathbf{f}$ is an $(h \times w \times d_e)$-dimensional feature map and $\mathbf{z}^l_{\text{occ}}$ is $(h \times w \times 1)$-dimensional binary output, but $\mathbf{z}_{\text{app}}$ is $(1 \times 1 \times d_p)$-dimensional feature vector. If $\sum_l \mathbf{z}^l_{\text{occ}} > 1$, the part occurs at multiple locations in an image. Since all these locations correspond to same part, their appearance should be the same. To incorporate this, we take the weighted average of the part appearance feature at each location, weighted by the probability that the part is present. Since we use the probability values for averaging the result is deterministic. This operation is encapsulated by the Q^{A} encoder (refer to Figure 2.2b). During image generation, we sample $\hat{\mathbf{z}}_{\text{app}}$ once and replicate it at each location where $\hat{\mathbf{z}}^l_{\text{occ}} = 1$. During training, this forces the model to: (1) only predict $\hat{\mathbf{z}}^l_{\text{occ}} = 1$ where similar looking parts occur, and (2) learn a common representation for the part that occurs at these locations. Note that $\mathbf{z}_{\text{app}}$ can be modeled as a mixture of distributions (e.g., mixture of gaussians) to capture complicated appearances. However, in this work we assume that the convolutional neural network based encoders are powerful enough to map variable appearance of semantic concepts to similar feature representations. Therefore, we restrict ourselves to a single gaussian distribution.

2.2.3 PatchVAE with multiple parts

Next we extend the framework described above to use multiple parts. To use N parts, we use $N \times 2$ encoder networks $Q^{\text{A}(i)}\left(\mathbf{z}^{(i)}_{\text{app}} \mid \mathbf{x}\right)$ and $Q^{\text{O}(i)}\left(\mathbf{z}^{l(i)}_{\text{occ}} \mid \mathbf{x}\right)$, where $\mathbf{z}^{(i)}_{\text{app}}$ and $\mathbf{z}^{l(i)}_{\text{occ}}$ parame-

terize the i^{th} part. Again, this can be implemented efficiently as 2 networks by concatenating the outputs together. The image generator samples $\hat{\mathbf{z}}_{\text{app}}^{(i)}$ and $\hat{\mathbf{z}}_{\text{occ}}^{l(i)}$ from the outputs of these encoder networks and constructs $\hat{\mathbf{z}}^{(i)}$. We obtain the final **patch latent code** $\hat{\mathbf{z}}$ by concatenating all $\hat{\mathbf{z}}^{(i)}$ in channel dimension. Therefore, $\hat{\mathbf{z}}^{(i)}$ is $(h \times w \times d_p)$-dimensional and $\hat{\mathbf{z}}$ is $(h \times w \times (N \times d_p))$-dimensional stochastic feature map. For this multiple part case, Equation (2.6) can be written as:

$$\begin{aligned} \mathbf{z_P} &= \left[\mathbf{z}_{\text{p}}^{(1)};\, \mathbf{z}_{\text{p}}^{(1)};\, \ldots;\, \mathbf{z}_{\text{p}}^{(N)}\right] \\ \text{where } \mathbf{z}_{\text{p}}^{(i)} &= \left[\mathbf{z}_{\text{occ}}^{1(i)};\, \mathbf{z}_{\text{occ}}^{2(i)};\, \ldots;\, \mathbf{z}_{\text{occ}}^{L(i)};\, \mathbf{z}_{\text{app}}^{(i)}\right]. \end{aligned} \tag{2.10}$$

Similarly, Equation (2.8) and Equation (2.9) can be written as:

$$\begin{aligned} &\mathcal{L}_{\text{MultiPatchVAE}}(\mathbf{x}) = -\mathbb{E}_{\mathbf{z_P}}\left[G\left(\mathbf{x} \mid \mathbf{z_P}\right)\right] \\ &+ \beta_{\text{app}} \sum_{i=1}^{N} \sum_{l=1}^{L} D_{\text{KL}}\left(Q^{\text{O}(i)}\left(\mathbf{z}_{\text{occ}}^{l(i)} \mid \mathbf{x}\right) \parallel \text{Bern}\left(\mathbf{z}_{\text{occ}}^{\text{prior}}\right)\right) \\ &+ \beta_{\text{occ}} \sum_{i=1}^{N} D_{\text{KL}}\left(Q^{\text{A}(i)}\left(\mathbf{z}_{\text{app}}^{(i)} \mid \mathbf{x}\right) \parallel \mathcal{N}\left(0, \mathcal{I}\right)\right) \end{aligned} \tag{2.11}$$

The training details and assumptions of posteriors follow the previous section.

2.2.4 Improved Reconstruction Loss

The L2 reconstruction loss used for training β-VAEs (and other reconstruction based approaches) gives equal importance to each region of an image. This might be reasonable for tasks like image compression and image de-noising. However, for the purposes of learning semantic representations, not all regions are equally important. For example, "sky" and "walls" occupy

large portions of an image, whereas concepts like "windows," "wheels,", "faces" are comparatively smaller, but arguably more important. To incorporate this intuition, we use a simple and intuitive strategy to weigh the regions in an image in proportion to the gradient energy in the region. More concretely, we compute laplacian of an image to get the intensity of gradients per-pixel and average the gradient magnitudes in 8×8 local patches. The weight multiplier for the reconstruction loss of each 8×8 patch in the image is proportional to the average magnitude of the patch. All weights are normalized to sum to one. We refer to this as **weighted loss** ($\mathcal{L}_{\text{w}}$). Note that this is similar to the gradient-energy biased sampling of mid-level patches used in [20, 21]. Examples of weight masks are provided in the supplemental material.

In addition, we also consider an adversarial training strategy from GANs to train VAEs [40], where the discriminator network from GAN implicitly learns to compare images and gives a more abstract reconstruction error for the VAE. We refer to this variant by using 'GAN' suffix in experiments. In Section 2.3.2, we demonstrate that the proposed weighted loss ($\mathcal{L}_{\text{w}}$) is complementary to the discriminator loss from adversarial training, and these losses result in better recognition capabilities for both β-VAE and PatchVAE.

2.3 Experiments

Datasets. We evaluate PatchVAE on CIFAR100 [25], MIT Indoor Scene Recognition [26], Places [27] and Imagenet [28] datasets. CIFAR100 consists of 60k 32×32 color images from 100 classes, with 600 images per class. There are 50000 training images and 10000 test images. Indoor dataset contains 67 categories, and a total of 15620 images. Train and test subsets consist of 80 and 20 images per class respectively. Places dataset has 2.5 millions of images with 205

categories. Imagenet dataset has over a million images from 1000 categories.

Learning paradigm. In order to evaluate the utility of PatchVAE features for recognition, we setup the learning paradigm as follows: we will first train the model in an unsupervised manner on all training images. After that, we discard the generator network and use only part of the encoder network $\phi(\mathbf{x})$ to train a supervised model on the classification task of the respective dataset. We study different training strategies for the classification stage as discussed later.

Training details. In all experiments, we use the following architectures. For CIFAR100, Indoor67, and Place205, $\phi(\mathbf{x})$ has a conv layer followed by two residual blocks [55]. For ImageNet, $\phi(\mathbf{x})$ is a ResNet18 model (a conv layer followed by four residual blocks). For all datasets, Q^{A} and Q^{O} have a single conv layer each. For classification, we start from $\phi(\mathbf{x})$, and add a fully-connected layer with 512 hidden units and a final fully-connected layer as classifier. More details can be found in the supplemental material.

During the unsupervised learning phase of training, all methods are trained for 90 epochs for CIFAR100 and Indoor67, 2 epochs for Places205, and 30 epochs for ImageNet dataset. All methods use ADAM optimizer for training, with initial learning rate of 1×10^{-4} and a minibatch size of 128. For relaxed bernoulli in Q^{O}, we start with the temperature of 1.0 with an annealing rate of 3×10^{-5} (following the details in [54]). For training the classifier, all methods use stochastic gradient descent (SGD) with momentum with a minibatch size of 128. Initial learning rate is 1×10^{-2} and we reduce it by a factor of 10 every 30 epochs. All experiments are trained for 90 epochs for CIFAR100 and Indoor67, 5 epochs for Places205, and 30 epochs for ImageNet datasets.

Table 2.1: Classification results on CIFAR100, Indoor67, and Places205. We initialize the classification model with the representations $\phi(\mathbf{x})$ learned from unsupervised learning task. The model $\phi(\mathbf{x})$ comprises of a conv layer followed by two residual blocks (each having 2 conv layers). First column (called 'Conv1') corresponds to Top-1 classification accuracy with pre-trained model with the first conv layer frozen, second and third columns correspond to results with first three and first five conv layers frozen respectively. Details in Section 2.3.1.

	CIFAR100			Indoor67			Places205		
Model	Conv1	Conv[1-3]	Conv[1-5]	Conv1	Conv[1-3]	Conv[1-5]	Conv1	Conv[1-3]	Conv[1-5]
β-VAE	44.12	39.65	28.57	20.08	17.76	13.06	28.29	24.34	8.89
β-VAE + $\mathcal{L}_{\text{w}}$	44.96	40.30	28.33	21.34	19.48	13.96	29.43	24.93	9.41
β-VAE-GAN	44.69	40.13	29.89	19.10	17.84	13.06	28.48	24.51	9.72
β-VAE-GAN + $\mathcal{L}_{\text{w}}$	45.61	41.35	31.53	20.45	18.36	14.33	29.63	25.26	10.66
PatchVAE	43.07	38.58	28.72	20.97	19.18	13.43	28.63	24.95	11.09
PatchVAE + $\mathcal{L}_{\text{w}}$	43.75	40.37	30.55	23.21	21.87	15.45	29.39	26.29	12.07
PatchVAE-GAN	44.45	40.57	31.74	21.12	19.63	14.55	28.87	25.25	12.21
PatchVAE-GAN + $\mathcal{L}_{\text{w}}$	45.39	41.74	32.65	22.46	21.87	16.42	29.36	26.30	13.39
BiGAN	47.72	41.89	31.58	21.64	17.09	9.70	30.06	25.11	10.82
Imagenet Pretrained	55.99	54.99	54.36	45.90	45.82	40.90	37.08	36.46	31.26

2.3.1 Downstream classification performance

In Table 2.1, we report the top-1 classification results on CIFAR100, Indoor67, and Places205 datasets for all methods with different training strategies for classification. First, we keep all the pre-trained weights in $\phi(\mathbf{x})$ from the unsupervised task frozen and only train the two newly added conv layers in the classification network (reported under column 'Conv[1-5]'). We notice that our method (with different losses) generally outperforms the β-VAE counterpart by a healthy margin. This shows that the representations learned by PatchVAE framework are better for recognition compared to β-VAEs. Moreover, better reconstruction losses ('GAN' and $\mathcal{L}_{\text{w}}$) generally improve both β-VAE and PatchVAE, and are complementary to each other.

Next, we fine-tune the last residual block along with the two conv layers ('Conv[1-3]' column). We observe that PatchVAE performs better than VAE under all settings except the for

Table 2.2: ImageNet classification results using ResNet18. We initialize weights from using the unsupervised task and fine-tune the last two residual blocks. Details in Section 2.3.1.

Model	Top-1 Acc.	Top-5 Acc.
β-VAE	44.45	69.67
PatchVAE	47.01	71.71
β-VAE + $\mathcal{L}_{\text{w}}$	47.28	71.78
PatchVAE + $\mathcal{L}_{\text{w}}$	47.87	72.49
Imagenet Supervised	61.37	83.79

CIFAR100 with just L2 loss. However, when using better reconstruction losses, the performance of PatchVAE improves over β-VAE. Similarly, we fine-tune all but the first conv layer and report the results in 'Conv1' column. Again, we notice similar trends, where our method generally performs better than β-VAE on Indoor67 and Places205 dataset, but β-VAE performs better CIFAR100 by a small margin. When compared to BiGAN, PatchVAE representations are better on all datasets ('Conv[1-5]') by a huge margin. However, when fine-tuning the pre-trained weights, BiGAN performs better on two out of four datasets. We also report results using pre-trained weights in $\phi(\mathbf{x})$ using *supervised* ImageNet classification task (last column, Table 2.1) for completeness. The results indicate that PatchVAE learns better semantic representations compared to β-VAE.

ImageNet Results. Finally, we report results on the large-scale ImageNet benchmark in Table 2.2. For these experiments, we use ResNet18 [55] architecture for all methods. All weights are first learned using the unsupervised tasks. Then, we fine-tune the last two residual blocks and train the two newly added conv layers in the classification network (therefore, first conv layer and the following two residual blocks are frozen). We notice that PatchVAE framework outperforms β-VAE under all settings, and the proposed weighted loss helps both approaches. Finally, the last row in Table 2.2 reports classification results of same architecture randomly initialized and

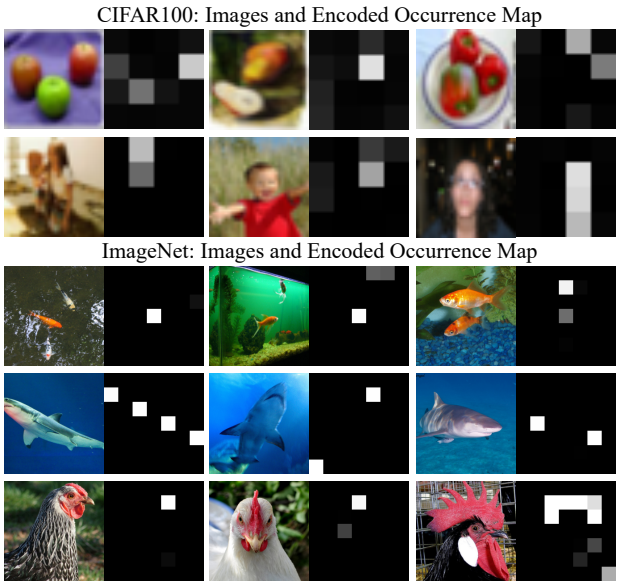


Figure 2.3: Encoded part occurrence maps discovered on CIFAR100 and ImageNet. Each row represents a different part.

trained end-to-end on ImageNet using supervised training for comparison.

Baselines. We use the β**-VAE** model (Section 2.2.1) as our primary baseline. In addition, we use weighted loss and discriminator loss resulting in the β**-VAE-*** family of baselines. We also compare against a **BiGAN** model from [39]. We use similar backbone architectures for encoder/decoder (and discriminator if present) across all methods, and tried to keep the number of parameters in different approaches comparable to the best of our ability. Exact architecture details can be found in the supplemental material.

2.3.2 Qualitative Results

We present qualitative results to validate our hypothesis. First, we visualize whether the structure we impose on the VAE bottleneck is able to capture occurrence and appearance of important parts of images. We visualize the PatchVAE trained on images from CIFAR100 and Imagenet datasets in the following ways.

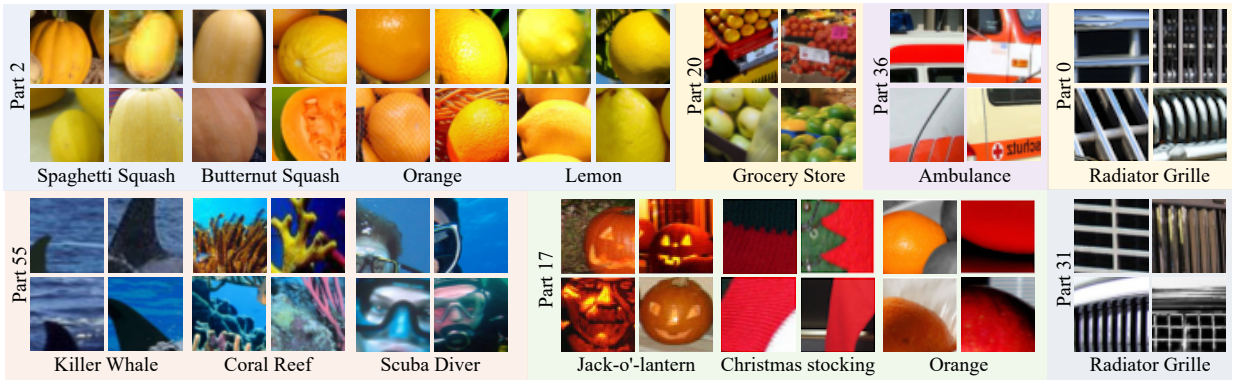


Figure 2.4: A few representative examples for several parts to qualitatively demonstrate the visual concepts captured by PatchVAE. For each part, we crop image patches centered on the part location where it is predicted to be present. Selected patches are sorted by part occurrence probability as score. We manually select a diverse set from the top-50 occurrences from the training images. As can be seen, a single part may capture diverse set of concepts that are similar in shape or texture or occur in similar context, but belong to different categories. We show which categories the patches come from (note that category information was not used while training the model).

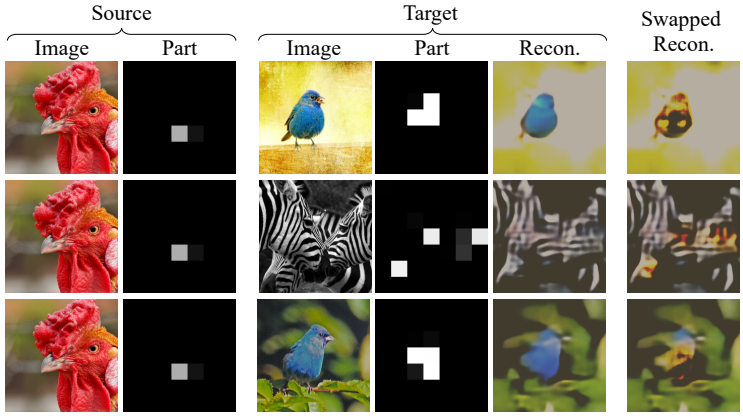


Figure 2.5: Swapping source and target part appearance. Column 1, 2 show a source image with the occurrence map of one of the parts. We can swap the appearance vector of this part with appearance vectors of a different part in target images. Column 3, 4 show three target images with occurrence maps of one of their parts. Observe the change in reconstructions (column 5, 6) as we bring in the new appearance vector. The new reconstruction inherits properties of the source at specific locations in the target.

Concepts captured. First, we visualize the part occurrences in Figure 2.3. We can see that the parts can capture round (fruit-like) shapes in the top row and faces in the second row regardless of the class of the image. Similarly for ImageNet, occurrence map of a specific part in images of chicken focuses on head and neck. Note that these semantically these parts are more informative than just texture or color what a β-VAE can capture. In Figure 2.4, we show parts captured by

the ImageNet model by cropping a part of image centered around the occurring part. We can see that parts are able to capture multiple concepts, similar in either shape, texture, or context in which they occur.

Swapping appearances. Using PatchVAE, we can swap appearance of a part with the appearance vector of another part from a different image. In Figure 2.5, keeping the occurrence map same for a target image, we modify the appearance of a randomly chosen part and observe the change in reconstructed image. We notice that given the same source part, the decoder tries similar things across different target images. However, the reconstructions are worse since the decoder has never encountered this particular combination of part appearance before.

Discriminative *vs*.Generative strength. As per our design, PatchVAE compromises the generative capabilities to learn more discriminative features. To quantify this, we use the the images reconstructed from β-VAE and PatchVAE models (trained on ImageNet) and compute three different metrics to measure the quality of reconstructions of test images. Table 2.7 shows that β-VAE is better at reconstruction.

2.4 Ablation Analysis

We study the impact of various hyper-parameters used in our experiments. For the purpose of this evaluation, we follow a similar approach as in the 'Conv[1-5]' column of Table 2.1 and all hyperparameters from the previous section. We use CIFAR100 and Indoor67 datasets for ablation analysis.

Maximum number of patches. Maximum number of parts N used in our framework. Depending on the dataset, higher value of N can provide wider pool of patches to pick from. However, it

Table 2.3: Effect of N: Increasing the maximum number of patches increases the discriminative power for CIFAR100 but has little or negative effect for Indoor67

N	CIFAR100	Indoor67
4	27.59	14.40
8	28.74	12.69
16	28.94	14.33
32	27.78	13.28
64	29.00	12.76

Table 2.4: Effect of d_p: Increasing the number of hidden units for a patch has very little impact on classification performance.

d_p	CIFAR100	Indoor67
3	28.63	14.25
6	28.97	14.55
9	28.21	14.55

Table 2.5: Effect of $\mathbf{z}_{\text{occ}}^{\text{prior}}$: Increasing increasing the prior probability of patch occurrence has adverse effect on classification performance.

$\mathbf{z}_{\text{occ}}^{\text{prior}}$	CIFAR100	Indoor67
0.01	28.86	14.33
0.05	28.67	14.25
0.1	28.31	14.03

Table 2.6: Effect of β_{occ}: Too high or too low β_{occ} can deteriorate the performance of learned representations.

β_{occ}	CIFAR100	Indoor67
0.06	30.11	14.10
0.3	30.37	15.67
0.6	28.90	13.51

Table 2.7: Reconstruction metrics on ImageNet. PatchVAE sacrifices reconstruction quality to learn discriminative parts, resulting in higher recognition performance (Table 2.2)

Model	PSNR ↑	FID ↓	SSIM ↑
β-VAE	4.857	108.741	0.289
PatchVAE	4.342	113.692	0.235

can also make the unsupervised learning task harder, since in a minibatch of images, we might not get too many repeat patches. Table 2.3(left) shows the effect of N on CIFAR100 and Indoor67 datasets. We observe that while increasing number of patches improves the discriminative power in case of CIFAR100, it has little or negative effect in case of Indoor67. A possible reason for this decline in performance can be smaller size of the dataset (i.e., fewer images to learn).

Number of hidden units for a patch appearance $\hat{\mathbf{z}}_{\text{app}}$. Next, we study the impact of the number of channels in the appearance feature $\hat{\mathbf{z}}_{\text{app}}$ for each patch (d_p). This parameter reflects the capacity of individual patch's latent representation. While this parameter impacts the reconstruction

quality of images. We observed that it has little or no effect on the classification performance of the base features. Results are summarized in Table 2.4(right) for both CIFAR100 and Indoor67 datasets.

Prior probability for patch occurrence $\mathbf{z}_{\mathbf{occ}}^{\mathbf{prior}}$. In all our experiments, prior probability for a patch is fixed to $1/N$, i.e., inverse of maximum number of patches. The intuition is to encourage each location on occurrence maps to fire for at most one patch. Increasing this patch occurrence prior will allow all patches to fire at the same location. While this would make the reconstruction task easier, it will become harder for individual patches to capture anything meaningful. Table 2.5 shows the deterioration of classification performance on increasing $\mathbf{z}_{\text{occ}}^{\text{prior}}$.

Patch occurrence loss weight $\beta_{\mathbf{occ}}$. The weight for patch occurrence KL Divergence has to be chosen carefully. If β_{occ} is too low, more patches can fire at same location and this harms the the learning capability of patches; and if β_{occ} is too high, decoder will not receive any patches to reconstruct from and both reconstruction and classification will suffer. Table 2.6 summarizes the impact of varying β_{occ}.

Chapter 3: Learning to Signal Mid-level Patches in Referential Games

The ability to communicate using language is a signature characteristic of intelligence [56]. Language provides a structured platform for agents to not only collaborate with each other and accomplish certain goals, but also to represent and store information in a compressible manner. Most importantly, language allows us to build infinitely many new concepts by the composition of the known concepts. These qualities are shared by both the natural languages used in human-human communication and programming languages used in human-machine communication. The study of the evolution of language can hence give us insights into intelligent machines that can communicate [57].

Our goal in this paper is to develop and understand an emergent language, *i.e.*, a language that emerges when two neural network agents try to communicate with each other. Clark [58] argued that supervised approaches that consist of a single agent learning statistical relationships among symbols don't capture the functional relationships between the symbols *i.e.*, the use of symbols leading to an action or an outcome. Krishna et al. [59] argued the same viewpoint in the context of images. We, therefore, resort to the recent works in emergent language [60, 61, 62, 63, 64, 65] which show that a communication protocol can be developed or learned by two or more cooperative agents trying to solve a task. The choice of the task is quintessential since the language derives meaning from its use [66]. We choose a task where two agents, a

speaker, and a listener, play a *referential game*, a type of signaling game first proposed by Lewis [67]. The speaker agent receives a target image and sends a message to the listener. The message consists of discrete symbols or words, capturing different parts of the image. The listener receives another view of the target image, and one or more distractor images. The goal of the speaker and the listener agents is to maximize the agreement between the message and the target image. Figure 3.1 illustrates the overview of the proposed referential game.

In computer vision, a number of attempts have been made to represent images as visual words [68], with a focus on low-level feature descriptors such as SIFT [69], SURF [70], *etc*.. Recent works in deep learning have attempted to describe the entire image with a fixed number of discrete symbols [71, 72, 73], however, we postulate that large images contain a lot of redundant information and a good visual representation should focus on only the "interesting" parts of the image. To discover what constitutes the interesting part of the image, we take inspiration from the works on **mid-level patches** [22, 74, 75], the patches in the image that are both *representative* and *discriminative* [12, 74]. This means they can be discovered in a large number of images (and hence representative), but simultaneously they should also be discriminative enough to set an image apart from the other images in the dataset. Hence, the speaker agent in our paper focuses on computing a symbolic representation in terms of these mid-level patches, as opposed to the entire image.

To summarize, we propose PatchGame, a referential game formulation where given an image, the speaker sends discrete signal in terms of mid-level patches, and the listener embeds these symbols to match them with another view of the same image in the presence of distractors. Compared to previous works [63, 64, 76], we make the following key changes:

- Agents in the some of the prior works [63, 64, 76] have access to a pre-trained network, such as AlexNet [77] or VGG [78], for extracting features from images. In this work, the agents rely on training on a large scale image dataset, and invariance introduced by various image augmentations, to learn the language in a self-supervised way [79].
- We propose a novel patch-based architecture for the speaker agent, which comprises of two modules: (1) PatchSymbol, a multi-layered perceptron (MLP) that operates at the patch-level and converts a given image patch into a sequence of discrete symbols, and (2) PatchRank, a ConvNet that looks at the complete image and ranks the importance of patches in a differentiable manner.
- We introduce a novel transformer-based architecture for the listener agent, consisting of two modules: (1) a language module that projects the message received from the speaker to a latent space, and (2) a vision module that projects the image into the latent space. We use a contrastive loss in this latent space to train both the speaker and the listener agents simultaneously.
- We propose new protocols to evaluate each of the speaker and listeners' modules.

We assess the success of PatchGame via qualitative and quantitative evaluations of each of the proposed component, and by demonstrating some practical applications. First, we show that the speaker's PatchRank model does indicate important patches in the image. We use the top patches indicated by this model to classify ImageNet [28] images using a pre-trained Vision Transformer [8] and show that we can retain over 60% top-1 accuracy with just half of the image patches. Second, the listener's vision model (ResNet-18) can achieve upto 30.3% Top-1 accuracy just by using k-NN ($k = 20$) classification. This outperforms other state-of-the-art unsupervised

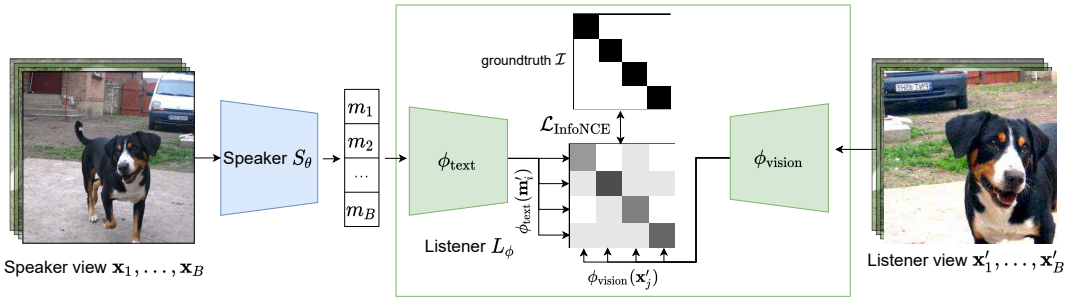


Figure 3.1: Overview of the Referential Game. We generate two random views of every image in the given batch of images. The speaker takes one of the views as the input and generates a sequence of symbols (or message). The listener takes the message sent by the speaker and the second view of the image, and projects both into an embedding space. Both the speaker and listener agents learn by minimizing the constrastive loss (see Equation (3.3)) between between the views in this embedding space.

approaches [12, 73] that learn discrete representations of images by 9%. Finally, we also analyze the symbols learned by our model and the impact of choosing several hyperparameters used in our experiments.

3.1 Related Work

Referential games. Prior to the advent of deep learning, significant research in the field of emergent communications has shown that a communication protocol can be developed or learned by agents by playing a language game [60, 61, 62, 80, 81, 82, 83, 84]. However, the agents employed in these works were typically located in a synthetic world and made several assumptions about the world such as the availability of disentangled representations of objects with discrete properties. More recent works [63, 65, 85, 86, 87, 88, 89, 90] have employed deep learning methods to develop a discrete language for communication between the agents. Lazaridou et al. [63] used neural network agents represented by a MLP to communicate concepts about real-world pictures. They used a fixed-sized message composed of a large vocabulary for their communication. Havrylov and Titov [64], Evtimova et al. [76], Bouchacourt and Baroni

[91] relax this assumption and allow communication via variable-length sequences. Havrylov and Titov [64] allows the speaker agent to use an LSTM [92] to construct a variable-length message. Havrylov and Titov [64], Lazaridou et al. [65] show that even when we allow agents to use variable-length sequences to represent a message, they tend to utilize the maximum possible sequence to achieve the best performance (in terms of communication success).

The idea of using a Gumbel-softmax distribution [53, 93] with the straight-through trick [94] for learning a language in multi-agent environment was concurrently proposed by Mordatch and Abbeel [95] and Havrylov and Titov [64]. They show that we can achieve a more stable and faster training by using this technique as compared to reinforcement learning used in several other approaches.

Evaluating emergent communication. Evaluating the emergent language turns out to be an equally challenging research problem. Existing approaches use the successful completion of the task or the correlation between learned language and semantic labels as evaluation metrics. Lowe et al. [96] and Keresztury and Bruni [97] show that simple task success might not be a good or sufficient metric for evaluating the success of a game. They discuss heuristics and advocate measuring both positive signaling and positive listening independently to evaluate agents' communication. Andreas [98] provides a way of evaluating compositional structure in learned representations.

Discrete Autoencoders. In parallel to the works on emergent communication, there is a large body of research on learning discrete representations of images using some form of autoencoding or reconstruction [12, 71, 73, 99, 100, 101, 102, 103] without labels. The focus of VQ-VAE [71] and VQ-VAE-2 [73] is to learn a discrete bottleneck using vector quantization. Once we can represent any image with these discrete symbols, a powerful generative model such

as PixelCNN [71, 104], or transformer [105, 106] is learned on top of these symbols to sample new images. PatchVAE [12] achieves the same using Gumbel-Softmax and imposes an additional structure in the bottleneck of VAEs. We argue that because of mismatch in the objectives of reconstruction and visual recognition tasks, each of these models trained using reconstruction-based losses do not capture meaningful representations in the symbols.

Self-supervised learning in vision. Self-supervised learning (SSL) methods, such as [107, 108, 109, 110, 111, 112, 113, 114] have shown impressive results in recent years on downstream tasks of classification and object detection. Even though the bottleneck in these methods is continuous (and not discrete symbols), these methods have been shown to capture semantic and spatial information of the contents of the image. Unlike SSL methods, neural networks representing the two agents in our case, do not share any weights. Also, note that the continuous nature of representations learned by SSL techniques is fundamentally different from the symbolic representations used in language. And indeed, we show that a k-Nearest Neighbor classifier obtained from the continuous representations learned by SSL methods can perform better than the one obtained using Bag of Words (or symbols). However, to the best of our knowledge, our work is one of the first attempts to make representations learned in a self-supervised way more communication- or language-oriented.

Comparison with Mihai and Hare [79, 115]. [79, 115] extends Havrylov and Titov [64] by training a speaker and listener agents end-to-end without using pre-trained networks. However, the prior works [64, 115, 115] use a **top-down** approach to generate discrete representation (or a sentence) for an image, i.e., they compute an overall continuous embedding of the image and then proceed by generating one symbol of the sentence at a time using an LSTM. The computational cost of LSTMs is prohibitive when length of a sentence is large, which is needed to

describe complex images. The transformers, on the other hand, require constant time for variable length sentences at the cost of increased memory (in the listener agent). However, generating the variable length sentences with the speaker agent using transformers is non-trivial. To solve this, we propose a **bottom-up** approach, *i.e.*, we first generate symbols for image patches and combine them to form a sentence. This approach allows for computationally efficient end-to-end training. Further, it allows the speaker to compose symbols corresponding to different parts of the image, instead of deducing it from a pooled 1D representation of the image.

3.2 Our Approach

We first introduce various notations and the referential game played by the agents in our work. We provide further details of architectures of the different neural network agents, as well as the loss function. We also highlight the important differences between this work and prior literature. Code and pre-trained models to reproduce the results are provided.

3.2.1 Referential Game Setup

Figure 3.1 shows the overview of our referential game setup. Given a dataset of N images $\{\mathbf{x}_i\}_{i=1}^{N}$, we formulate a referential game [67] played between two agents, a speaker S_θ and a listener L_ϕ as follows: As in the setting of Grill et al. [114], we generate two "random views" for every image. A random view is generated by taking a 224×224 crop from a randomly resized image and adding one or more of these augmentations - color jitter, horizontal flip, Gaussian blur and/or solarization. This prevents the neural networks from learning a trivial solution and encourages the emergent language to capture invariances induced by the augmentations. Given

a batch of B images, we refer to the two views as $\{\mathbf{x}_i\}_{i=1}^{B}$ and $\{\mathbf{x}'_i\}_{i=1}^{B}$. In each iteration during training, one set of views is presented to the speaker agent and another set of the views is shown to the listener agent.

The speaker S_θ encodes each image $\mathbf{x}_i$ independently into a variable length message $\mathbf{m}_i$. Each message $\mathbf{m}$ is represented by a sequence of one-hot encoded symbols with a maximum possible length L and a fixed size vocabulary V. The space of all possible messages sent by the speaker is of the order $|V|^L$. The input to the listener L_ϕ is the batch of messages $\{\mathbf{m}_i\}_{i=1}^{B}$ from the speaker, and the second set of random views of the batch of images. The listener consists of a language module ϕ_{text} to encode messages and a vision module ϕ_{vision} to encode images. The goal of the listener is to match each message to its corresponding image.

Specifically, for a batch of B (message, image) pairs, S_θ and L_ϕ are jointly trained to maximize the cosine similarities of B actual pairs while minimizing the similarity of $(B^2 - B)$ incorrect pairs. For a target message $\mathbf{m}_j$, the image $\mathbf{x}'_j$ (augmented view of $\mathbf{x}_j$) acts as the target image while all the other $(B-1)$ images act as distractors. And vice versa, for the image $\mathbf{x}'_k$, the message $\mathbf{m}_k$ (encoded by the speaker $\mathrm{S}_\theta(\mathbf{x}_k)$) acts as the target message while all the other $(B-1)$ messages act as distractors. We use the following symmetric and contrastive loss function, also sometimes referred to as InfoNCE loss in previous metric-learning works [72, 116].

$$\mathcal{L}_{\text{text}} = -\sum_{j=1}^{B} \log \frac{\exp(\phi_{\text{text}}(\mathbf{m}_j) \cdot \phi_{\text{vision}}(\mathbf{x}'_j)/\tau)}{\sum_{k=1}^{B} \exp(\phi_{\text{text}}(\mathbf{m}_j) \cdot \phi_{\text{vision}}(\mathbf{x}'_k)/\tau)} \tag{3.1}$$

$$\mathcal{L}_{\text{vision}} = -\sum_{k=1}^{B} \log \frac{\exp(\phi_{\text{text}}(\mathbf{m}_k) \cdot \phi_{\text{vision}}(\mathbf{x}'_k)/\tau)}{\sum_{j=1}^{B} \exp(\phi_{\text{text}}(\mathbf{m}_j) \cdot \phi_{\text{vision}}(\mathbf{x}'_k)/\tau)} \tag{3.2}$$

$$\mathcal{L} = (\mathcal{L}_{\text{text}} + \mathcal{L}_{\text{vision}})/2 \tag{3.3}$$

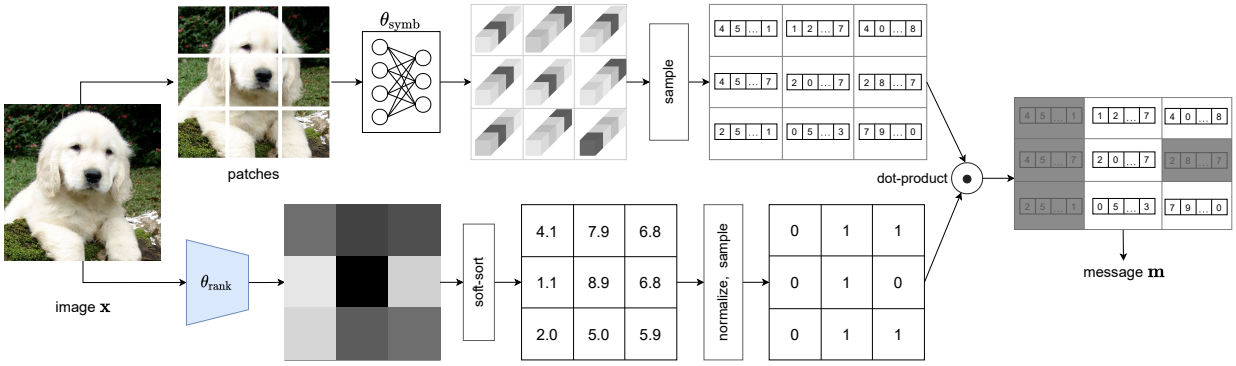


Figure 3.2: Speaker agent architecture. The upper branch represents the PatchSymbol, θ_{symb} module and the lower branch represents PatchRank, θ_{rank} module. Each of the modules take the raw image as the input. PatchSymbol computes a message for each patch of pre-defined size, using a fixed size vocabulary and message length. PatchRank uses a ConvNet to compute weights or the importance of each patch. Note that the symbols for a patch are context independent, however the importance of a patch depends on the context. The speaker agent combines the output of each of the models and sends a variable length message to the listener.

where τ is a constant temperature hyperparameter.

The game setting used in our work is inspired from Lazaridou et al. [63] and Havrylov and Titov [64], but there are important differences. Both our speaker S_θ and listener L_ϕ agents are trained from scratch. This makes the game setting more challenging, since agents cannot use the pre-trained models which have been shown to encode semantic and/or syntactic information present in natural language or images. Our training paradigm, where we show different views of the same image to the speaker and listener, is inspired by the recent success of self-supervised learning in computer vision. Empirically, we observe that this leads to a more stable training and prevents the neural networks from learning degenerate solutions. However, in contrast with such self-supervised approaches, our goal is to learn a discrete emergent language as opposed to continuous semantic representations. We discuss the differences in the architecture of the two agents in the following sections.

3.2.2 Speaker agent architecture

A desirable characteristic of the speaker agent is that it should be able to encode "important" components of images with a variable length sequence of discrete symbols. Previous works [64, 76, 91] have achieved this by first converting the image into a continuous deterministic latent vector and then using an LSTM network [92] to generate a sequence of hidden states, and sample from this sequence of hidden state until a special end of sequence token (or maximum length) is reached. As observed by [64, 65], in order to achieve the minimum loss, the model ends up always using the maximum allowable length. In our experiments as well, we observed that having an LSTM makes the training slow and does not achieve the objective of encoding images in variable length sequences. We propose leverage two separate modules in the speaker agent S_θ to circumvent this problem - the first module called PatchSymbol (θ_{symb}) is a 2-layer MLP that computes patch-level embeddings for the image, the second module called PatchRank (θ_{rank}) is a small ConvNet that computes rank or importance of each patch in the image.

PatchSymbol, θ_{symb}. The idea of encoding an image at the patch level is inspired by the works on discovering mid-level patches [22, 74, 75] in images. We use a simple 2-hidden layer MLP, to encode each $\mathbb{R}^{C\times S^2}$ dimensional image patch $\mathbf{x}_{\text{patch}}$ to l vectors of log of probabilities $[\log p_1^1, \ldots, \log p_V^1], \ldots, [\log p_1^l, \ldots, \log p_V^l]$. Here C is the number of (color) channels in the input image or patch, S is the spatial dimension of a square patch, V is the size of the vocabulary used to represent a single symbol, and l is the number of symbols used to encode each patch. Hence an image of size $\mathbb{R}^{C\times H\times W}$ can be encoded using $K = \frac{HW}{S^2}$ patches, each consisting of l symbols. The vectors of log probabilities allow us to sample from a categorical distribution of V categories, with a continuous relaxation by using the Gumbel-softmax trick [53, 93]. For a given

vector $[\log p_1^j, \ldots, \log p_V^j]$, we draw i.i.d samples g_i^j from the $\mathsf{Gumbel}(0, 1)$ distribution [117] and get a differentiable approximation of $\arg\max$ as follows:

$$\left[[\log p_1^1, \ldots, \log p_V^1], \ldots, [\log p_1^l, \ldots, \log p_V^l]\right] = \text{MLP}(\mathbf{x}_{\text{patch}}) \tag{3.4}$$

$$y^j = \mathsf{one\_hot}(\arg\max_i [\log p_i^j]) \sim \left\{ \frac{\exp((\log p_i^j + g_i^j)/\tau_s))}{\sum_{k=1}^{V} \exp((\log p_k^j + g_k^j)/\tau_s)} \right\}_{i=1}^{V}, \forall j \tag{3.5}$$

$$\theta_{\text{symb}}(\mathbf{x}_{\text{patch}}) = \mathbf{y} = \left\{y^j\right\}_{j=1}^{l} \tag{3.6}$$

where τ_s controls how close the approximation is to $\arg\max$. The final output of the θ_{symb} network for the entire image $\mathbf{x}$ is L one-hot encoded $V-$dimensional symbols, where $L = l \times \frac{HW}{S^2}$. In all our experiments, we fix $V = 128$ and $l = 1$.

PatchRank, θ_{rank}. An image might have a lot of redundant patches encoded using the same symbols. The goal of the θ_{rank} network is to give an importance score to each patch. Since importance of a patch depends on the context and not the patch alone, we use a small ResNet-9 [55] to compute an importance weight for each of the $K = \frac{HW}{S^2}$ patches. One possible way to use these importance weights is to simply normalize them between $(0, 1)$ and repeat the Gumbel-Softmax trick to sample important patches. The listener network L_ϕ would see only the message consisting of "important" patches. However, we empirically observed that a simple min-max or L2-normalization allows the network to assign high weights to each patch and effectively send the entire sequence of length L to the listener. Instead, we propose to use a *differentiable ranking* algorithm by Blondel et al. [118] to convert the importance weights to soft-ranks $\{1, \ldots, K\}$ in $O(K \log K)$ time. This method works by constructing differentiable operators as projections onto the convex hull of permutations. Once we have the vector of soft-ranks $\mathbf{r} \in \mathbb{R}^K$, we normalize

the ranks and sample binary values again using a special case of the Gumbel-softmax trick for Bernoulli distributions [53, 93] as

$$\mathbf{r}(\mathbf{x}) = \arg\max_{\pi \in \Sigma} \langle \text{CNN}(\mathbf{x}), \rho_\pi \rangle \tag{3.7}$$

$$\theta_{\text{rank}}(\mathbf{x}) \sim \text{Bern}\left(\frac{1}{K}\mathbf{r}(\mathbf{x})\right) \tag{3.8}$$

where $\text{CNN}(\mathbf{x})$ are the importance weights obtained by applying a ResNet-9 to the image $\mathbf{x}$, Σ is the set of all $K!$ permutations, and ρ_π are the ranks corresponding to the permutation $\pi \in \Sigma$. We refer the reader to [118] for a detailed description of the soft-sort algorithm. Therefore, the final symbols encoded by the speaker agent, S_θ, is given by:

$$\text{S}_\theta(\mathbf{x}) = \mathbf{m} = \theta_{\text{symb}}(\mathbf{x}) \cdot \theta_{\text{rank}}(\mathbf{x}) \tag{3.9}$$

3.2.3 Listener agent architecture

As discussed in Section 3.2.1, the listener agent L_ϕ consists of a language module ϕ_{text} and a vision module ϕ_{vision}. We implement ϕ_{text} using a small transformer encoder [105]. We prepend a $\langle\text{CLS}\rangle$ token at the beginning of each message sequence received from the speaker [103, 119], and use the final embedding of $\langle\text{CLS}\rangle$ to compute the loss described in Equation (3.3). We implement ϕ_{vision} using a small vision transformer [8], and follow a similar procedure as in the text module to obtain the final image embedding. Both the text and vision modules use a similar transformer encoder architecture (no weight sharing) with 192 hidden size, 12 layers and 3 attention heads. Following [107, 108, 111], we add a high dimensional projection at the end of the last layer

before computing the loss function.

3.2.4 Training

Each of the weights of the speaker and listener agents $\{\theta_{\text{symb}}, \theta_{\text{rank}}, \phi_{\text{vision}}, \phi_{\text{text}}\}$ are optimized jointly during the training. We use a 2-layer MLP for θ_{symb}, ResNet-9 [55] for θ_{rank}, a ResNet-18 [55] for ϕ_{vision}, and a small transformer encoder (hidden size $= 192$, 3 heads, 12 layers) for ϕ_{text}. All the experiments are conducted on the training set of ImageNet [28], which has approximately 1.28 million images from 1000 classes. We create a training and validation split from the training set by leaving aside 5% of the images for validation. After obtaining the final set of hyper-parameters, we retrain on the entire training set for 100 epochs. We use Stochastic Gradient Descent (SGD) with momentum and cosine learning rate scheduling. In order to train the stochastic component of the Speaker, we use the straight-through trick [94]. We reiterate that the speaker and listener do not share weights, and the only supervision used is the InfoNCE loss defined in Equation (3.3). Please refer to the appendix and code attached in the supplementary material for more details.

3.3 Experiments

We evaluate the success of communication in the referential game and impact of various hyper-parameters on the success. Also, following the work of Lowe et al. [96], we evaluate the emergent communication in two primary ways. In section 4.1, we measure *positive signaling*, which means that S_θ sends messages relevant to the observation. In section 4.2, we measure *positive listening*, which indicates that the messages are influencing the L_ϕ agent's behavior.

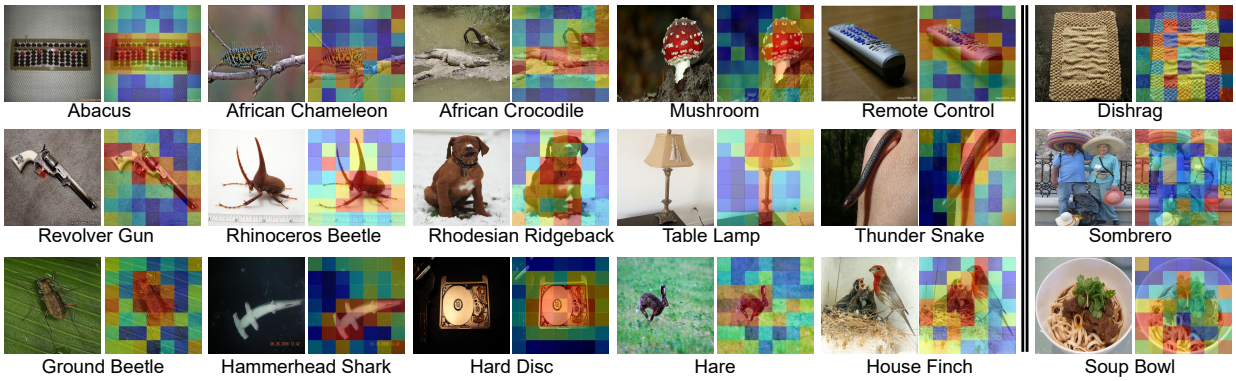


Figure 3.3: Heat maps based on Patch Importance. Red represents the most important patches and Blue the least important. Labels are listed for better understanding and are not used during training. We have used 224×224 images and 32×32 patches to get 49 non-overlapping patches per image. Model successfully separates the primary object in an image - the "device" in the "Abacus" image, "rabbit" in the "hare" image etc. *(as shown by higher concentration of yellow-to–red patches on these areas).* Some failure cases are shown as well on the right.

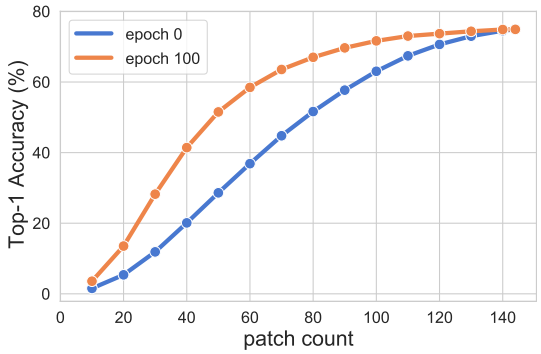
3.3.1 Positive Signaling - Visualizing Patch Ranks from θ_{rank}

We first visualize the output of PatchRank module as a heatmap overlayed on the original image in Figure 3.3. Most important patches are colored towards 'red' and the least important ones are colored towards 'blue'. The figure shows that our PatchRank can capture important and discriminative parts of the images. In case of images of various animals and plants, the model assigns the highest important to discriminative body parts. For the inanimate objects such as abacus or revolver the model is able to distinguish between the foreground and the background. Note that although approaches such as GradCam [120] can provide pixel-level importance heatmaps, they require extensive supervision. Our method on the other hand is self-supervised. We also show some of the failure cases when discriminative patches in the image cover majority of the image on the rightmost 2 columns of Figure 3.3.

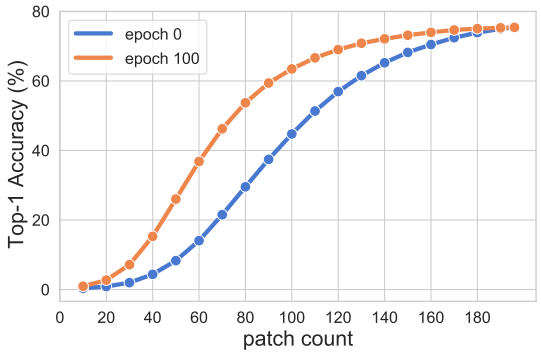
3.3.2 Positive Signaling - Image Classification with subset of patches provided by θ_{rank}

Recently proposed Vision Transformers (ViT) by Dosovitskiy et al. [8] have gained popularity because of their simplicity and performance. These models treat an image as a sequence of $N \times N$ patches, use an MLP to convert the patch into an embedding, and finally use these set of patches to perform classification. We first note that both the inference time and memory consumption of ViT depend largely on the length of sequence, because of the $O(N^2)$ self-attention operation. Secondly, the performance of the Vision Transformers drops if instead of using all patches, we only use a subset of patches during inference. This provides us a simple way to evaluate the θ_{rank} module. We artificially constrain the number of patches available to ViT during inference. We measure the Top-1 Accuracy of ViT using only k allowed patches. The selection of the patches is done by the θ_{rank} model.

We consider two different pre-trained ViT [8] models, one trained using 32×32 patches, on the images of size 384×384 (so the number of patches in an image is 144), while the second model is trained using 16×16 patches on the images of size 224×224 (196 patches per image). In Figure 3.4a and Figure 3.4b, we show the Top-1 accuracy obtained by the pre-trained ViT models using important patches predicted by θ_{rank} at different values of k. At the 0^{th} epoch (with random weights), the performance of ViT drops almost linearly as we lower the patch count k. At the 100^{th} epoch, at the end of the training, we observe that performance of the ViT does not drop as drastically.



(a) 32×32 patches



(b) 16×16 patches

Figure 3.4: Impact of number of patches used during evaluation in a pre-trained ViT [8]. The performance of a ViT drops if we provide fewer patches during inference. However a PatchRank model (which is a small ResNet-9) can provide important patches to ViT during inference with minimal loss in Top-1 accuracy.

3.3.3 Positive Signaling - Visualizing θ_{symb} symbols

As mentioned earlier, we are using a vocabulary $V = 128$ and patch size $S = 32$ in our base model. This means θ_{symb} has to map each 32×32 patch to one of the 128 available symbols. A natural way to analyze what the symbols are encoding to see is to visualize their corresponding patches. While $V = 128$ is far too small a vocabulary to describe all possible patches, we observe some interesting patterns in Figure 3.5. Many of the symbols seem to adhere to specific concepts repeatedly. We observed that symbols have a lesser preference for color, but more preference for texture and shape. We discovered several symbols corresponding to textures such as grass, branches, and wood. We also noticed many symbols firing for the patches corresponding to text, eyes, and even faces. There can be multiple symbols representing single concept, *e.g.*, both symbol 91 and 123 both fire in case of eyes.

Figure 3.5: Visualizing patches corresponding to various symbols. The number in the top row corresponds to 1 of 128 vocabulary ids. 6 representative patches corresponding to each patch are shown. The bottom row corresponds to our interpretation of what concept that symbol might be capturing.

Table 3.1: Performance of Listener's Vision module ϕ_{vision} - Downstream classification accuracy for ImageNet dataset using k-NN (k=20)

Method	Top-1 (%)	Top-5 (%)
MoCo-v2 [113] (R18)	36.8 ($\pm$ 0.2)	60.3
VQ-VAE2 [72]	17.2 ($\pm$ 0.6)	30.5
PatchVAE [12] (R18, $S = 16$)	16.4 ($\pm$ 0.6)	28.5
PatchVAE [12] (R18, $S = 32$)	21.3 ($\pm$ 0.5)	36.2
Ours (R18, $S = 16$)	<u>27.6</u> ($\pm$ 0.6)	<u>46.2</u>
Ours (R18, $S = 32$)	**30.3** ($\pm$ 0.5)	**49.9**

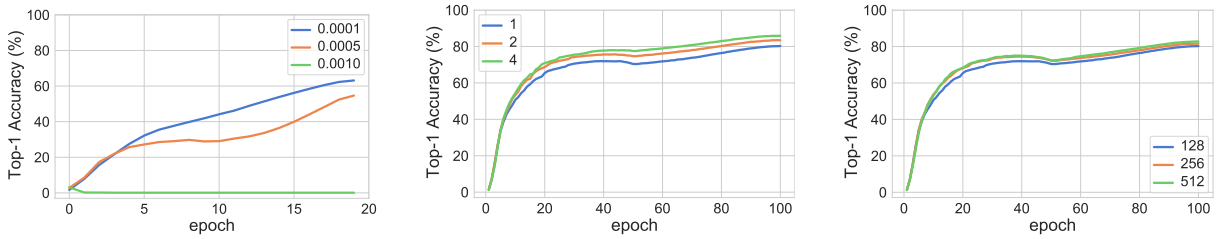
Table 3.2: Performance of Listener's Vision module ϕ_{vision} - Downstream mean Average Precision for Pascal VOC

Method	mAP-50 (%)
MoCo-v2 [113] (R18)	65.8
PatchVAE [12] ($S = 16$)	52.2
PatchVAE [12] ($S = 32$)	54.2
Ours ($S = 16$)	<u>58.9</u>
Ours ($S = 32$)	**61.3**

3.3.4 Positive Listening

Next, we evaluate the vision module, or ϕ_{vision} of the listener. We follow the protocol employed by various approaches in self-supervised learning literature. We consider the features obtained at the final pooling layer of the ϕ_{vision} (which is a ResNet-18 in our case). Next, we run a k-NN classification with $k = 20$ on the validation dataset. Table 3.1 shows the Top-1 and Top-5 % accuracy obtained on ImageNet using the listener's vision module and the baselines approaches. Although our method outperforms VQ-VAE-2 and PatchVAE (methods that learn a discrete image representation) we observe that there is still a gap between representations learned by these models as compared to the representations learned by continuous latent models such as MoCo [113]. Note that, because of the resource constraints, all results reported in the table are obtained by training ResNet-18 for only 100 epochs. The results for both MoCo-v2 and our ap-

proach continue to improve if we continue the training beyond 100 epochs (as also noted by He et al. [113]). Further, we use the listener's vision module as a pre-trained network for Pascal VOC dataset [121]. Our results are shown in the Table 3.2. Again, results are not competitive as compared to self-supervised counterparts such as MoCo-v2 but we outperform models with discrete bottleneck such as PatchVAE. We find that the convergence of models with discrete bottlenecks (such as this work, and PatchVAE) is slow and hence, improving the training efficiency of this class of models is an interesting future direction.



(a) Learning rate (b) Length of a single patch (c) Vocabulary

Figure 3.6: Impact of various hyperparameters on the success of communication. Top-1 accuracy denotes the percentage of messages in a batch that model was able to match to the corresponding image. Having a higher message length and vocabulary helps the model to learn faster.

3.4 Ablation study

A communication iteration is successful if the receiver is able to match the message sent by the speaker to the correct target image. In our experiments, we use an effective batch size of 512 (split over 4 GPUs), so the chance accuracy of success of the speaker is 0.19%. We measure the Top-1 accuracy for each image of the batch and average it over validation data at the end of epoch. In Figure 3.6a, we observe that too high or too low learning rates can be detrimental to the success of communication. We fix the learning rate at 0.0001 which shows the best validation performance empirically. We train our models at different vocabulary sizes, and

different message lengths for 100 epochs. From Figure 3.6b and Figure 3.6c, we observe that having either a large vocabulary or larger message length allows the model to reach high accuracy faster. This is intuitive since the larger the space spanned by the messages is, the easier it is for receiver to distinguish between the images. A large message length increases the possible number of messages exponentially, at the cost of much larger computation cost since self-attention [105, 119] is an $O(N^2)$ operation. A large vocabulary also increases the both the span of messages and computation cost linearly.

Chapter 4: Aligning Sparse in-the-wild Image Collections

Given an image of a car, we as humans, can easily map corresponding pixels between this car and an arbitrary collection of car images. Our visual system is able to achieve this (rather impressive) feat using a multitude of cues - low level photometric consistency, high level visual grouping and our priors on cars as an object category (shape, pose, materials, illumination etc.). The above is also true for an image of a "never-before-seen" object (as opposed to a common object category such as cars) where humans demonstrate surprisingly robust generalization despite lacking an object or category specific priors [122]. These correspondences in turn inform downstream inferences about the object such as shape, affordances, and more. In this work, we tackle this problem of "low-shot dense correspondence" – *i.e.*given only a small in-the-wild image collection ($\sim$10–30 images) of an object or object category, we recover dense and consistent correspondences across the entire collection.

Prior works addressing this problem of dense alignment in "in-the-wild" image collections assume availability of annotated keypoint matches and image pairs [123, 124], a mesh of the object [125, 126], or a very large collection of images of the object [127, 128]. These assumptions often do not hold for the long-tail of objects that exist in real world imagery. This long-tail is unavoidable; no matter how many new images we annotate, we will keep uncovering new and rare categories of objects. In our work, we show that it is possible to achieve dense correspondence

of small in-the-wild image collections without any manual annotations by leveraging the power of large self-supervised vision models. Aligning these image sets can be useful for a wide range of applications such as edit propagation for images and videos, as well as downstream problems such as pose and shape estimation.

In the presence of a limited number of samples and a high-dimensional search space, dense correspondence and joint alignment of an image set is a challenging optimization problem. We draw inspiration from classical image alignment methods [129, 130] where images are warped (or congealed) to a consistent canonical pose before classification using simple transformations, as well as recent works on per-image-set optimization [131, 132, 133], where the inductive model biases coupled with additional regularization allows for learning a good solution with self-supervision. Our framework, dubbed ASIC, consists of a small image-to-image network which predicts a dense per-pixel mapping from the image to a two-dimensional canonical grid. This canonical grid is parameterized as a multi-channel learned embedding and stores an RGB color along with an alpha value indicating whether the location represents the object or the background. We devise a novel contrastive loss function to ensure that semantic keypoints from different images map to a consistent location in canonical space.

The key contribution of our work is to exploit noisy and sparse pseudo-correspondences between a pair of images and extend them to learn consistent dense correspondences across the image collection. These pseudo-correspondences can be obtained using any of the large self-supervised learning (SSL) models [109, 110, 111, 112, 113, 114, 134] which learn without explicit labels on large internet-scale data. In order to make them accurate, we enforce pairwise consistency across the image collection with an alignment network and a self-supervised keypoint consistency loss. Further, we introduce additional regularization via equivariance and

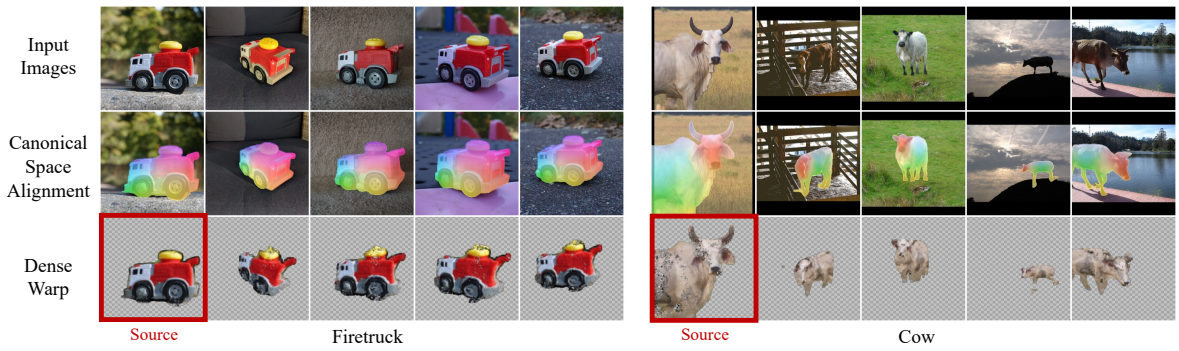


Figure 4.1: **Globally consistent and dense aligments with ASIC.** Given a small set (∼10-30) of images of an object or object category captured in-the-wild, our framework computes a dense and consistent mapping between all the images in a self-supervised manner. **First row:** Unaligned sets of images from the SAMURAI (Keywest) and SPair-71k (Cow) datasets. **Second row:** Dense correspondence maps produced by our method. **Third row:** Image in the first column warped to the images in columns 2-5.

reconstruction terms to get dense correspondences across collection.

Figure 4.1 demonstrates the dense and consistent mapping learned by our model for two image sets. We also evaluate our method on 18 image categories in SPair-71k [135] dataset, 4 categories in PF-Willow [136], 3 fine-grained categories in the CUB [137], as well as 5 collections in SAMURAI [138] datasets and show that ASIC is competitive against unsupervised keypoint correspondence approaches, and often outperforms them. An additional advantage of learning a joint canonical mapping is that our method suffers significantly less drift when propagating keypoints on a sequence of images (instead of just a single image pair). In order to evaluate the keypoint consistency over a sequence of k images, we propose a new metric k-CyPCK (or k-cycle PCK) in Sec. 4.3.4 and show that our method outperforms existing methods by over 20% at both the low and high precision settings. In summary, our contributions are as follows:

- We introduce a test-time optimization technique to recover consistent dense correspondence maps over a small collection of in-the-wild images.
- We design a novel loss function and several regularization terms to encourage mapping to be

consistent across multiple images in a given collection.

- We perform extensive quantitative and qualitative evaluations on 4 different datasets (spanning 30 object categories) to show that our method is competitive with the unsupervised methods, often outperforming them.
- We propose a novel metric k-CyPCK to evaluate consistency of keypoint propagation over a sequence of images, which is not captured by traditional metrics such as PCK.

4.1 Related Work

Correspondence between image pairs. Keypoint matching or correspondence between images is one of the oldest tasks in computer vision. Some very early works focused on finding dense optical flow [139, 140, 141] between pairs of consecutive images in videos via a variational framework to optimize flow based on pixel intensities. Sparse keypoint matching, e.g., using SIFT descriptors [142], also gained importance due to applications in tracking [143, 144] and structure from motion (SfM) [145, 146, 147]. SIFT Flow [148] proposed the idea of using SIFT descriptors for dense alignment between image pairs. Initial deep learning based correspondence works [149, 150, 151] replaced SIFT with deep features. With the availability of labeled datasets, a number of works have performed end-to-end matching with deep networks [123, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162]. However, a shortcoming of these aforementioned works is that they usually require large labeled datasets, and often fail to generalize on unseen objects or scenes.

Joint alignment of image sets. The concept of a canonical image has long been used for the task of object detection via template matching [163, 164]. Learned-Miller *et al*. [130, 165] formal-

ized the task of jointly aligning a set of images (i.e., congealing them) by continuously warping each image (*e.g.*via affine transformations) to minimize the entropy distribution of the image set. [166] use deep features from multiple resolutions in place of hand-crafted features. GANgealing [127] extended this idea by constraining the canonical image to be the output of a pre-trained StyleGAN [167, 168]. In a similar vein, CoordGAN [128] trains a structure-texture disentangled GAN with a canonical coordinate frame as input. Both of these works attempt to solve a similar tasks as ours, but are limited by data-hungry GAN training. Some works exploits 3D shape as a means for consistent dense correspondences across image collections [125, 169, 170, 171] but require access to additional signals such as category specific 3D templates, segmentation masks or keypoint correspondences. In contrast, our work attempts to learn dense correspondences in a low-shot setting where GAN training is infeasible and in the absence of additional training signals. As mentioned before, we do so primarily by leveraging large pre-trained SSL models as our source of semantic priors on general imagery.

Self-supervised correspondence discovery. To overcome the lack of large datasets with ground-truth correspondence, recent work seeks to combine the idea of distilling deep features from a network trained with self-supervision on large-scale image datasets. Some of these works optimize for proxy losses computed with known transformations [172, 173, 174, 175, 176, 177, 178, 179, 180]. Like these methods, we also train our network to be equivariant to synthetic geometric transformations. However, a key difference is that we also train with pseudo-correspondences 'across' real images, which allows the method to generalize better and build a consistent mapping across the given image collection.

Deep Matching Prior [181] and Neural Best Buddies [182] optimize for only a single

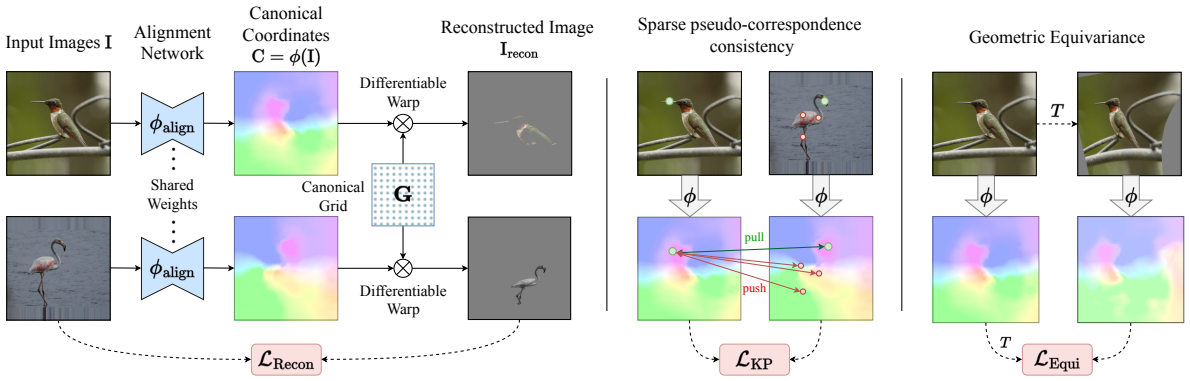


Figure 4.2: **ASIC Architecure.** The alignment network $\mathbf{\Phi}_{\text{align}}$ predicts canonical space coordinates for all pixels for all input images. Images can be reconstructed using a differentiable warp from the canonical space. In order to align semantically similar pixels from different images to the same location in the canonical space, we propose two primary loss functions $\mathcal{L}_{\text{KP}}$ and $\mathcal{L}_{\text{Recon}}$. Please refer to the Sec. 4.2.2 for more details.

pair of images to match deep features of one image to another. More recently PSCNet [183] and Neural Congealing [184] train large networks for simultaneously matching deep features for image pairs by learning a flow from image to image, and image to canonical space respectively. However, these methods have limited flexibility in the deformation space and do not generalize well to out of plane rotations present in datasets such as SPair-71k. We allow our model to map different image regions arbitrarily to different parts of the canonical space. In Tab. 4.1, we show that this allows us to generate more accurate correspondences and generalize to more object categories.

4.2 ASIC Framework

Given a collection of images of an object or an object category, our goal is to assign corresponding pixels in all the images to a unique location in a canonical space. By doing so, we can use this learned canonical space as an intermediary when mapping pixels from one image to any another image in the collection while guaranteeing global consistency. The absence of ground

truth annotations, small size of the datasets we consider ($\sim$10 - 30 images) and the presence of occlusions and variations in shape, texture, viewpoint, and background lighting all serve to make this task highly challenging. We introduce a simple yet robust framework with a novel self-supervised contrastive loss function over image pairs, as well as auxiliary regularization losses on this learned canonical space to find consistent dense correspondences across the collection.

4.2.1 Obtaining Pseudo-correspondences

Prior work has shown that deep features extracted from large pre-trained networks contain useful local semantic information [110, 185, 186]. In this work, we use DINO [110] to extract these local semantic features. Note that these features are only extracted for obtaining pseudo-correspondences only once and are not used during the training. Given a pair of images $\mathbf{I}^{\mathrm{a}}$ and $\mathbf{I}^{\mathrm{b}}$, we obtain feature maps $\mathbf{F}^{\mathrm{a}}$ and $\mathbf{F}^{\mathrm{b}}$ using DINO. Here $\mathbf{F}^{\mathrm{a}} = \{\mathbf{f}_i^{\mathrm{a}}\}$ and $\mathbf{F}^{\mathrm{b}} = \{\mathbf{f}_i^{\mathrm{b}}\}$ represents the sets of feature vectors $\mathbf{f} \in \mathbb{R}^{\mathrm{d}}$ for all spatial locations $\mathbf{p}_i = (x, y) \in \mathbb{R}^2$. In practice, we obtain these feature maps at a coarser resolution, but for brevity, we do not introduce new notations for low-resolution feature maps. We define our pseudo keypoint correspondences, between the two images $\mathbf{I}^{\mathrm{a}}$ and $\mathbf{I}^{\mathrm{b}}$ as all pairs of locations of feature vectors that are mutual nearest neighbors, *i.e.*,

$$\{(\mathbf{p}_i^{\mathrm{a}}, \mathbf{p}_j^{\mathrm{b}}) \mid \left(\mathrm{NN}(\mathbf{f}_i^{\mathrm{a}}, \mathbf{F}^{\mathrm{b}}) = \mathbf{f}_j^{\mathrm{b}}\right) \ \wedge \ \left(\mathrm{NN}(\mathbf{f}_j^{\mathrm{b}}, \mathbf{F}^{\mathrm{a}}) = \mathbf{f}_i^{\mathrm{a}}\right)\}$$

where $\mathrm{NN}(\mathbf{f}_i^{\mathrm{a}}, \mathbf{F}^{\mathrm{b}})$ corresponds to the nearest neighbor of the normalized feature vector $\mathbf{f}_i^{\mathrm{a}}$ in the set of feature vectors $\mathbf{F}^{\mathrm{b}}$. The mutual nearest neighbors are usually noisy and sparse, and they serve as pseudo-correspondences for training our alignment network which we discuss next.

4.2.2 Architecture

Fig. 4.2 gives the high-level overview of the framework. Formally, we are given an image collection consisting of N images $\{\mathbf{I}^k\}_{k=1}^N$. We want to train an alignment network $\mathbf{\Phi}_{\text{align}}$ that takes a single image as input at a time and outputs $\mathbf{C} = \mathbf{\Phi}_{\text{align}}(\mathbf{I})$, the canonical space coordinate map, of the image. The canonical space coordinate map $\mathbf{C}$ has the same spatial dimensions as the input $H \times W$ and contains (u, v) coordinates in the shared canonical grid for that location. We parameterize this alignment network $\mathbf{\Phi}_{\text{align}} : \mathbb{R}^{H\times W\times 3} \rightarrow \mathbb{R}^{H\times W\times 2}$ with a fully convolutional U-Net [187] trained from scratch for the collection. Each pixel location $\mathbf{p} = (x, y)$ of this map consists of a 2-dimensional $\mathbf{c} = (u, v)$ coordinate.

These coordinates corresponds to a location in a learned canonical grid $\mathbf{G} \in \mathbb{R}^{H'\times W'\times 4}$. The canonical grid $\mathbf{G}$ is also two-dimensional but can have arbitrary height and width $H' \times W'$, and is shared by all the images in the collection. Each location in canonical grid $\mathbf{G}$ stores an (r, g, b, α) value which corresponds to colors (r, g, b) and a probability α that this location corresponds to a foreground pixel in the image. The original image, and a foreground visibility mask can now be reconstructed using this shared canonical grid $\mathbf{G}$, canonical space mapping $\mathbf{C}$, and a differentiable warp operator commonly used in spatial transformer networks [188]. For the mapping to be meaningful, we want semantically similar points from different images to map to the same location in the canonical space. In the next section, we describe the training loss we devised to this end.

4.2.3 Training Objectives

Sparse pseudo-correspondence consistency. The central goal of our framework is to ensure that semantically similar points in the images are aligned in the canonical space. Recall from the Sec. 4.2.1, that we pre-compute the pseudo-correspondences between all pairs of images using mutual nearest neighbors in the SSL feature space. Since SSL models are not trained for the task of correspondence, the pseudo-correspondences are noisy and sparse. Our first loss term is targeted at improving the accuracy of the correspondences by jointly aligning them for all pairwise combinations of images in our collection. Formally, given an image pair $(\mathbf{I}^{\mathrm{a}}, \mathbf{I}^{\mathrm{b}})$, we denote all the K pseudo-correspondences in the pair by $\{\mathbf{p}_i^{\mathrm{a}}, \mathbf{p}_i^{\mathrm{b}}\}_{i=1}^{K}$. We apply the alignment network $\mathbf{\Phi}_{\text{align}}$ to $\mathbf{I}^{\mathrm{a}}$ and $\mathbf{I}^{\mathrm{b}}$ independently to obtain the canonical space coordinates for each pixel in the pair, which we denote by $\{\mathbf{c}_i^{\mathrm{a}}, \mathbf{c}_i^{\mathrm{b}}\}_{i=1}^{K}$. We want to map each keypoint location in $\mathbf{I}^{\mathrm{a}}$ as close as possible to its counterpart in $\mathbf{I}^{\mathrm{b}}$, while pushing it away from the mapping of other keypoints in $\mathbf{I}^{\mathrm{b}}$. To achieve this, we define our first loss function $\mathcal{L}_{\text{KP}}$ as

$$\mathcal{L}_{\text{KP}} = -\sum_{i=1}^{K} \log \frac{\exp(-\parallel \mathbf{c}_i^{\mathrm{a}} - \mathbf{c}_i^{\mathrm{b}} \parallel^2 / \tau)}{\sum_{j=1}^{K} \exp(-\parallel \mathbf{c}_i^{\mathrm{a}} - \mathbf{c}_j^{\mathrm{b}} \parallel^2 / \tau)} \tag{4.1}$$

where τ is a hyperparameter and is fixed to 1.0 in all our experiments. $\mathcal{L}_{\text{KP}}$ plays the key role in improving the accuracy of pseudo-correspondences jointly for all images in our collection. However, the number of pseudo-correspondences is still very small (typically 100-300 for an image pair) as compared to the number of pixel locations. Hence, this loss is sparse and we need to add extra regularization terms in order to learn dense alignment, that we will discuss next.

Geometric transformation equivariance. In order to make our learned mapping dense, we

introduce a geometric equivariance regularization term in our loss function. We apply a random synthetic geometric transformation $\mathcal{T}$ to a given image $\mathbf{I}$. Since the output of the alignment network $\mathbf{\Phi}_{\text{align}}$ learns the canonical space coordinates for each location of input image, we can apply the same geometric transformation $\mathcal{T}$ to $\mathbf{\Phi}_{\text{align}}(\mathbf{I})$, and enforce an equivariance loss as follows

$$\mathcal{L}_{\text{Equi}} = \| \mathcal{T}(\mathbf{\Phi}_{\text{align}}(\mathbf{I})) - \mathbf{\Phi}_{\text{align}}(\mathcal{T}(\mathbf{I})) \| \tag{4.2}$$

where $\mathcal{T}$ is the geometric transformation. We choose thin plate spline (TPS) transformations [189] in our work, commonly used for image warps. $\mathcal{L}_{\text{KP}}$ and $\mathcal{L}_{\text{Equi}}$ serve as the two primary loss functions for the image set alignment problem, serving the purpose of making the pseudo-correspondences accurate and dense respectively. To further aid the training, we also propose the following auxiliary regularizations.

Total variation regularization. In order to encourage smooth mappings from from each image to the canonical space, we add a total variation (TV) regularization to the computed mapping $\mathbf{C}$. We found TV loss to be crucial to mitigate degenerate solutions (see Sec. 4.4):

$$\mathcal{L}_{\text{TV}} = \mathcal{L}_{\text{Huber}}(\Delta_x(\mathbf{C} - \mathcal{I})) + \mathcal{L}_{\text{Huber}}(\Delta_y(\mathbf{C} - \mathcal{I})) \tag{4.3}$$

where $\mathcal{I}$ is the identity mapping (*i.e.*each pixel (x, y) in the image gets mapped to (x, y) in the canonical space), Δ_x and Δ_y denote the partial derivatives under finite differences w.r.t.x and y dimensions, and $\mathcal{L}_{\text{Huber}}$ denotes the Huber loss [190].

Reconstruction loss. All the loss terms so far are computed on the canonical coordinates given

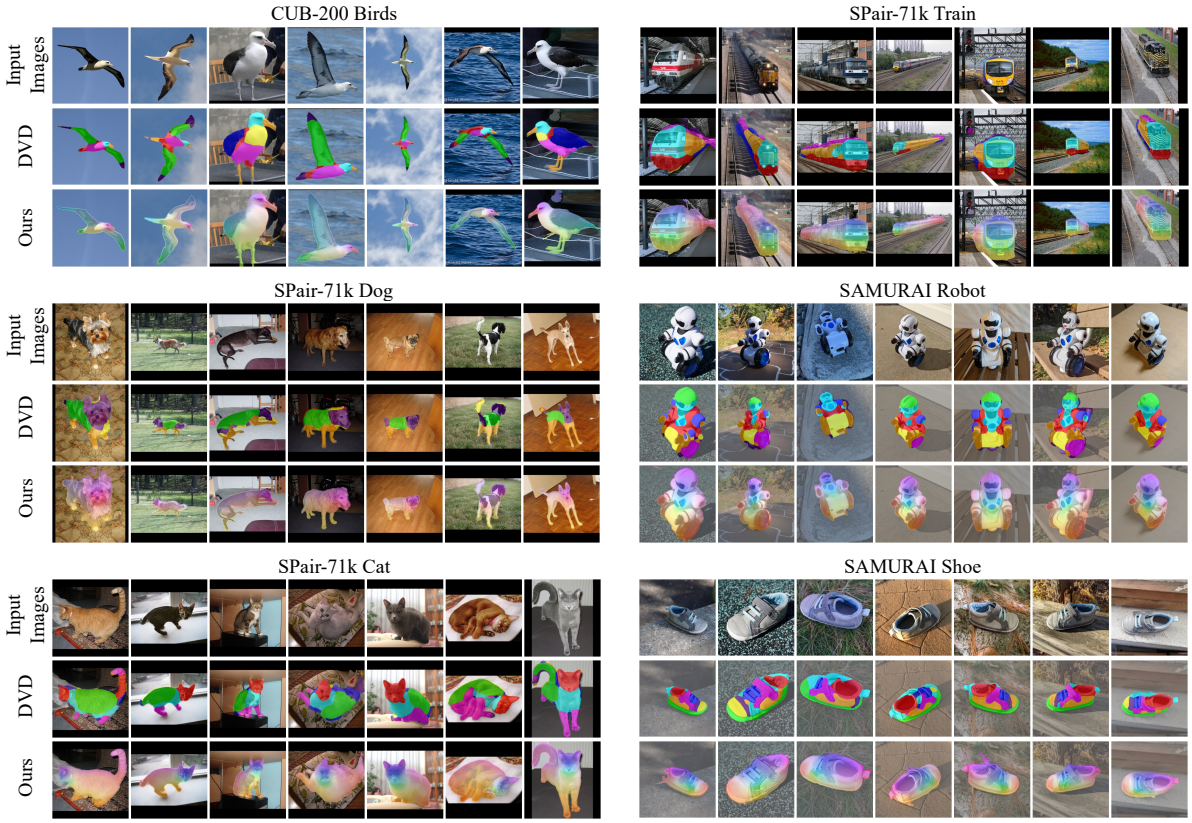


Figure 4.3: **Visualizing canonical space alignment.** For each dataset, the top row shows sample images from the dataset (composed of 10-30 images each). The middle row shows part co-segmentations computed by DVD [9]. DVD computes a coarse, discrete set of parts across the dataset. The bottom row shows the continuous canonical space mapping computed by our method. Our canonical space mapping is smooth and consistent across the images for each dataset/collection.

by $\mathbf{\Phi}_{\text{align}}$ and does not use the canonical grid $\mathbf{G}$. Recall that $\mathbf{G}$ is of the size $H' \times W' \times 4$, and allows us to reconstruct each image as well as a foreground visibility mask via a differentiable warp operator ($\mathcal{W}$) [188] such that $\mathbf{I}_{\text{Recon}}, \mathbf{M}_{\text{Recon}} = \mathcal{W}(\mathbf{G}, \mathbf{C})$. This allows us to compute a per image reconstruction loss using the original and reconstructed images. However a simple L_1 or L_2 loss will not suffice since $\mathbf{G}$ is shared for all the images in the collection and these images may come from wildly different backgrounds and lighting conditions. Furthermore, the two images might contain two different instances of the same object class, and may have different textures, shapes, and viewpoints. We instead minimize the perceptual (LPIPS) loss [191] which

measures a perceptual patch level similarity between images. For the reconstructed mask, we compute pixel-wise binary cross entropy (BCE) loss using the image foregrounds obtained with co-segmentation [9].

$$\mathbf{I}_{\text{Recon}}, \mathbf{M}_{\text{Recon}} = \mathcal{W}(\mathbf{G}, \mathbf{C})$$

$$\mathcal{L}_{\text{Recon}} = \text{LPIPS}(\mathbf{I}, \mathbf{I}_{\text{Recon}}) + \text{BCE}(\mathbf{M}, \mathbf{M}_{\text{Recon}}) \tag{4.4}$$

Consistent part alignment. For our final auxiliary loss, we obtain part co-segmentation maps from the images [9, 185] by clustering deep ViT features into S semantic parts, then running GrabCut [192] to smoothen the part boundaries. Our hypothesis is that semantically similar parts in the images should get mapped to similar location in $\mathbf{G}$, Formally, for each image $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$, we obtain semantic part masks as a binary matrix, $\mathbf{P} \in \mathbb{R}^{H \times W \times S}$. Since we want a part across the image set to map to a compact location in the canonical space, we minimize the variance of the canonical space coordinates for all pixels belonging to a part:

$$\mathcal{L}_{\text{Parts}} = \sum_{s=1}^{S} \frac{1}{N_s} \sum_{i} \parallel \mathbf{c}_i^s - \mathbb{E}(\mathbf{c}^s) \parallel^2 \tag{4.5}$$

where N_s is the number of pixels belonging to the part, $\mathbf{c}_i^s$ is the canonical coordinates of i^{th} pixel location belonging to the s^{th} part, and $\mathbb{E}(\mathbf{c}_i^s)$ is the centroid of the s^{th} part. We fix the number of parts to 8 in all our experiments. Alternately, the number of parts can be computed using the elbow method [193] at the expense of additional compute.

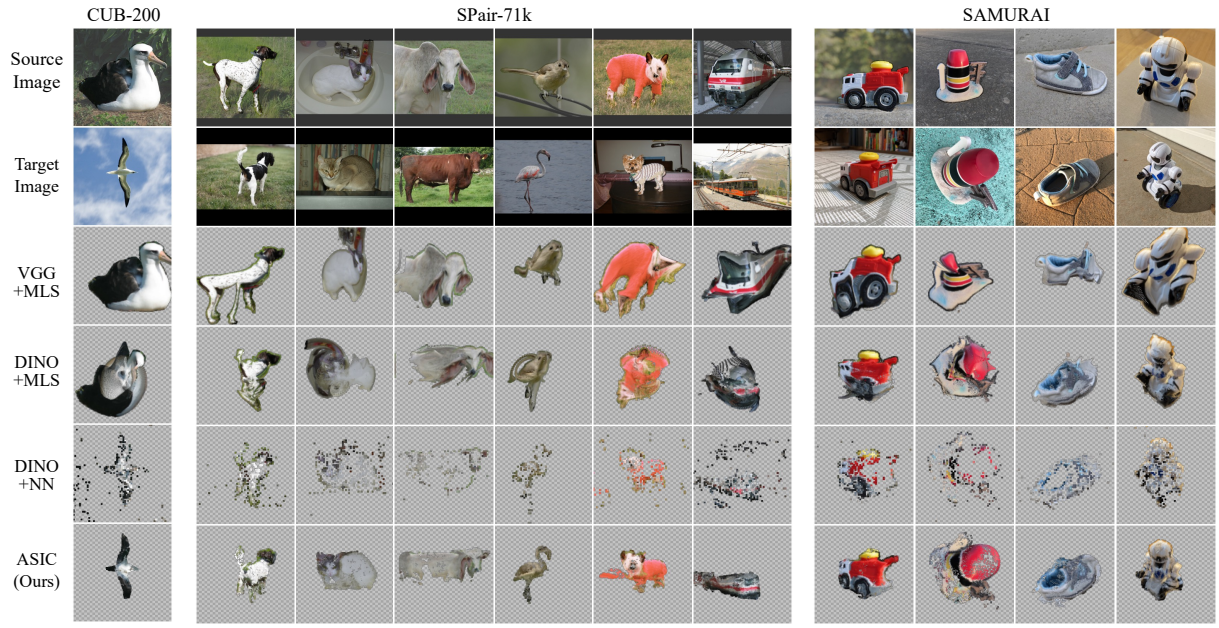


Figure 4.4: **Dense warping** from a source image (top row) to a target image (second row). We warp all foreground pixels (highlighted by a red overlay in the source image). Our methods produce dense and semantically more meaningful warps from the source to the target.

4.3 Experiments

We evaluate our method on several real-world in-the-wild image collections of both rigid and non-rigid object categories. For all datasets, we use a fixed set of hyperparameters (provided in the appendix) unless specified otherwise.

Datasets. SPair-71k [135] consists of 1,800 images from 18 categories. We optimize over image collections derived from the SPair-71k test set for each category independently and report results on each individual category, as well as aggregate results over all 18 categories. In case of **PF-Willow** [136], we consider all 4 categories of the dataset containing ∼30 images. **CUB-200** [137] datasets consists of over 200 fine-grained categories. We optimized our model on the test sets of first 3 categories of the dataset, consisting of 15-20 images each. We also show qualitative results on 4 objects from **SAMURAI** dataset [138].

4.3.1 Canonical Space Alignment

One simple way to visualize the alignment of an image set when mapped to the canonical space $\mathbf{G}$ is to define a colormap over the canonical grid and color the image pixels according to their mapped location in the canonical grid. In Fig. 4.3, the first row for each collection contains sample input images. The second row shows discrete parts obtained via parts co-segmentation using [9]. While these parts are also consistent across the image set, our canonical space mapping (third row) can be seen as a dense and continuous co-segmentation. We show the results for six datasets: CUB-200 birds; Dogs, Cats, and Train from SPair-71k; and Robot and Shoe from SAMURAI. The colormap used for the canonical space is provided in the supplement. We observe that our method can find dense correspondences across highly varying poses, backgrounds, and lighting. It also maps common parts of objects in a dataset to nearby regions of the canonical space. This is evident in Fig. 4.3 where, for instance, the faces of different cats are colored similarly.

4.3.2 Visualizing Dense Correspondences

We can also find dense correspondences between a pair of images $\mathbf{I}^a$ and $\mathbf{I}^b$ using our framework. Recall that $\mathbf{\Phi}_{\text{align}}$ outputs canonical space coordinates $\mathbf{c}^a$ and $\mathbf{c}^b$ for each pixel location $\mathbf{p}^a$ and $\mathbf{p}^b$. In order to warp the source image $\mathbf{I}^a$ to a target image $\mathbf{I}^b$, for every foreground pixel $\mathbf{p}_i^a$ in $\mathbf{I}^a$, we need to find its nearest neighbor among the set of points in $\mathbf{p}^b$ in the canonical space: We perform this action for all the foreground pixels in the source image, and splat according to the nearest neighbor mapping to get our desired warped image. Fig. 4.4 shows qualitative results for 10 different datasets. The top row is the source image with foreground mask highlighted.

Table 4.1: **Evaluation on SPair-71k.** Per-class and average PCK@0.10 on test split. Highest PCK among *weakly supervised* methods in bold, second highest underlined. Scores marked with ($\star$) means the paper uses a fixed image from the test set as canonical image. Our method is competitive against other weak supervised approaches and often outperforms them.

Supervision	**Method**	Aero	Bike	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Dog	Horse	Motor	Person	Plant	Sheep	Train	TV	**All**
Strong Supervision	SCorrSAN [162]	57.1	40.3	78.3	38.1	51.8	57.8	47.1	67.9	25.2	71.3	63.9	49.3	45.3	49.8	48.8	40.3	77.7	69.7	55.3
GAN supervision	GANgealing [127]	-	37.5	-	-	-	-	-	67.0	-	-	23.1	-	-	-	-	-	-	57.9	-
Weak supervision (train/test)	CNNGeo [195]	23.4	16.7	40.2	14.3	36.4	27.7	26.0	32.7	12.7	27.4	22.8	13.7	20.9	21.0	17.5	10.2	30.8	34.1	20.6
	A2Net [172]	22.6	18.5	42.0	16.4	37.9	**30.8**	26.5	35.6	13.3	29.6	24.3	16.0	21.6	22.8	**20.5**	13.5	31.4	36.5	22.3
	WeakAlign [179]	22.2	17.6	41.9	15.1	38.1	27.4	27.2	31.8	12.8	26.8	22.6	14.2	20.0	22.2	17.9	10.4	32.2	35.1	20.9
	NCNet [157]	17.9	12.2	32.1	11.7	29.0	19.9	16.1	39.2	9.9	23.9	18.8	15.7	17.4	15.9	14.8	9.6	24.2	31.1	20.1
	SFNet [196]	26.9	17.2	45.5	14.7	38.0	22.2	16.4	**55.3**	13.5	33.4	27.5	17.7	20.8	21.1	16.6	15.6	32.2	35.9	26.3
	PMD [153]	26.2	18.5	48.6	15.3	**38.0**	21.7	17.3	51.6	13.7	34.3	25.4	18.0	20.0	24.9	15.7	16.3	31.4	**38.1**	26.5
	PSCNet-SE [183]	28.3	17.7	45.1	15.1	37.5	30.1	27.5	47.4	14.6	32.5	26.4	17.7	24.9	24.5	19.9	16.9	34.2	37.9	27.0
Weak supervision (test-time optimization)	VGG+MLS [182]	29.5	22.7	61.9	**26.5**	20.6	25.4	14.1	23.7	14.2	27.6	30.0	29.1	24.7	27.4	19.1	19.3	24.4	22.6	27.4
	DINO+MLS [110, 182]	49.7	20.9	63.9	19.1	32.5	27.6	22.4	48.9	14.0	36.9	39.0	30.1	21.7	41.1	17.1	18.1	35.9	21.4	31.1
	DINO+NN [9]	57.2	24.1	67.4	24.5	26.8	29.0	27.1	52.1	15.7	42.4	43.3	30.1	23.2	40.7	16.6	24.1	31.0	24.9	33.3
	NeuCongeal [184]	-	**29.1***	-	-	-	-	-	53.3	-	-	35.2	-	-	-	-	-	-	-	-
	ASIC (Ours)	**57.9**	25.2	**68.1**	24.7	35.4	28.4	**30.9**	54.8	**21.6**	**45.0**	**47.2**	**39.9**	**26.2**	**48.8**	14.5	**24.5**	**49.0**	24.6	**36.9**

The second row is the target image. We show results for two other pairwise image optimization approaches, and then our method in the last row. NBB [182] computes nearest neighbors using VGG-19 and applies a Moving Least Squares (MLS) optimization [194] to compute a dense flow from source to target. While the flow computed via MLS is smooth, it usually does not respect semantic correspondences, as evident in the figure. We extend their technique to use DINO features as well. With DINO and the nearest neighbor approach (DVD) [9], the semantic correspondences are arguably better, but since this approach relies on the output of a Vision Transformer (ViT), which has lower resolution than the image, it produces a sparse flow. Our method produces both dense and consistent flow between an image pair.

4.3.3 Pairwise Correspondence

Metric. For evaluating accuracy of pairwise correspondence, we use the PCK metric [197] (percentage of correct keypoints) on the SPair, CUB, and PF-Willow datasets.

Table 4.2: **CUB-200 and PF-Willow.** PCK@0.10 for three CUB categories and four PF-Willow categories.

	CUB-200 (3 categories)		PF-Willow (4 categories)	
Method	PCK@0.1	PCK@0.05	PCK@0.1	PCK@0.05
PMD [153]	-	-	74.7	40.3
PSCNet-SE [183]	-	-	75.1	42.6
VGG+MLS [182]	25.8	18.3	63.2	41.2
DINO+MLS [110, 182]	67.0	52.0	66.5	45.0
DINO+NN [9]	68.3	52.8	60.1	40.1
ASIC (Ours)	**75.9**	**57.9**	**76.3**	**53.0**

Baselines. We categorize prior works based on the supervision used: (1) *Strong supervision* methods utilize human-annotated keypoints to learn pairwise image correspondence and achieve the best performance (on average). We include the numbers from a recent work [162] for reference purposes. (2) *GAN supervision* methods like [127] use a category-specific GAN pre-trained with large external datasets. While this method works well, it is restricted to only the categories for which large datasets are available and GAN training is feasible. (3) *Weak supervision* methods use category-level supervision (*i.e.*they assume that given pair/collection of images are from same category). They often resort to fine-tuning a large ImageNet [28] pre-trained network using a self-supervised loss function (*e.g.*with synthetic transformations) and optionally use additional information such as foreground masks or matching image pairs for training. Some of these works follow a *train/test* setting, where the network is fine-tuned on a separate set of training images. Note that in our work, we train a much smaller network from scratch instead of fine-tuning a large network. Some approaches (including ours) directly perform *test-time optimization* without additional training data or annotations.

NBB [182] optimizes a flow from one image to another using mutual nearest neighbors as control points [194]. While [182] shows the results by computing nearest neighbors from a

VGG network, we further extend their work to utilize a DINO network. [9] simply computes nearest neighbors in DINO feature space. A concurrent work, Neural Congealing [184], is closest to our work, in that they also perform test-time training using a canonical atlas. However, for objects with large deformations (such as in SPair-71k), they need to apply category specific accommodations (for instance, fixing the atlas for bicycle category). Our canonical grid allows for large deformations and is learned in all cases with a fixed set of hyperparameters. We obtain scores for other models from their respective papers (whenever available) or from [162]. Scores for [9, 110, 182] are computed using official code. The official code of [184] did not converge on several objects in our experiments, hence we report the quantitative results from the paper.

Discussion. Tab. 4.1 shows PCK@0.1 for all SPair-71k [135] categories. It is evident that having groundtruth keypoint annotations during training is highly beneficial; approaches that lack keypoints during training lag behind. We also observe that for categories with rigid objects (or less extreme deformations) such as 'Bottle' or 'Bus', weakly supervised approaches attain a similar performance as ours. However, in the objects with extreme variations such as animals/birds, our method outperforms other baselines. In our experiments, we observed that per-category hyperparameters can increase PCK performance further by $\sim 2\%$. This strategy is similar to Neural Congealing [184] where specific accommodations are made per category (*e.g.*tailored training regime for bicycle). However we report our numbers with a fixed set of hyperparameters for consistency.

Tab. 4.2 shows average results for the first 3 categories of the CUB dataset, and 4 categories of the PF-Willow dataset. Note that PF-Willow is an easier dataset compared to SPair-71k since it consists of rigid objects with little variation. Our method has performance similar to PSCNet-

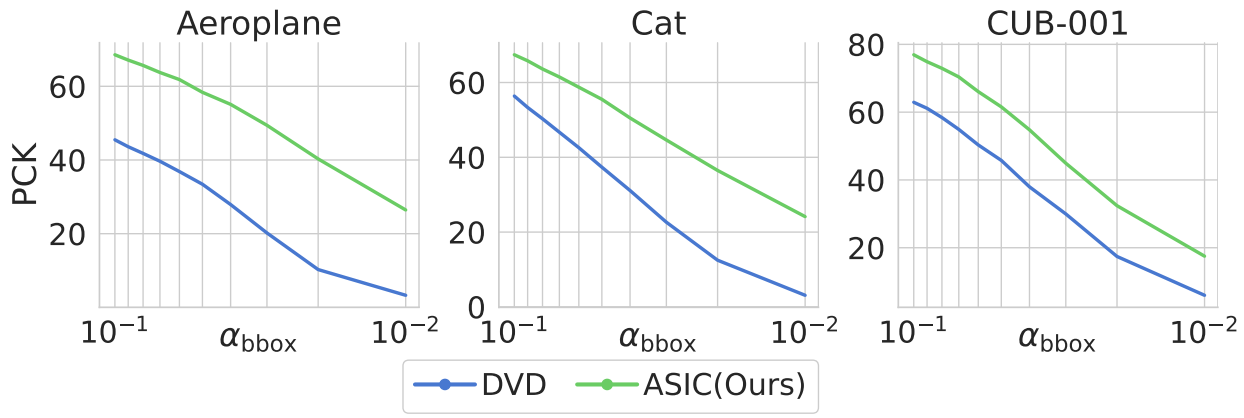


Figure 4.5: **Image Set Correspondence.** k-CyPCK at varying α_{bbox} (higher is better). Our method outperforms DVD baseline at both large and small values of α_{bbox}.

SE [183] when we compute PCK using threshold $\alpha_{\text{bbox}} = 0.1$ (which corresponds to a $\sim$20-pixels margin of error). However at higher precision ($\alpha_{\text{bbox}} = 0.05$), our method provides much larger gains compared to the baselines.

4.3.4 Image Set Correspondence

Our goal in this work is to recover dense and *consistent* correspondences. A shortcoming of the PCK metric is that it is only computed between image pairs. However, the errors in keypoint prediction tend to accumulate when transferring keypoints over a sequence of images. To address this limitation, we propose a new metric to measure consistency across multiple images, called k-CyPCK. Given a set of k images $\{\mathbf{I}^1, \mathbf{I}^2, \ldots, \mathbf{I}^k\}$ and an annotated keypoint in the first image $\mathbf{p}^1$ visible in **each** of the k images, we propagate $\mathbf{p}$ from $\mathbf{I}^1 \rightarrow \mathbf{I}^2, \mathbf{I}^2 \rightarrow \mathbf{I}^3, \ldots, \mathbf{I}^{k-1} \rightarrow \mathbf{I}^k, \mathbf{I}^k \rightarrow \mathbf{I}^1$ and get the corresponding predictions $\mathbf{p}^{1\rightarrow 2}, \mathbf{p}^{1\rightarrow 2\rightarrow 3}, \ldots$ and so on. As before, $\mathbf{p}^{1\rightarrow\cdots\rightarrow j}$ is considered to be predicted correctly if it is within a threshold $\alpha_{\text{bbox}} \cdot \max(H_{\text{bbox}}, W_{\text{bbox}})$ of the ground truth keypoints $\hat{\mathbf{p}}^{1\rightarrow\cdots\rightarrow j}$. We sum up all the correct predictions and plot scores at different values of α_{bbox} in Fig. 4.5. We choose $k = 4$ for all experiments (with additional results for other

values of k provided in the supplement). Note that since the number of possible permutations of k-length sequences can be very large, we randomly sample 200 sequences in our experiments.

Fig. 4.5 shows that our method significantly outperforms the DINO+NN baseline for both small and large values of α_{bbox} across all datasets (complete results in the supplemental material). We attribute this result to having a consistent canonical space across the image collection that prevents errors in keypoint transfer from accumulating to large values.

4.4 Ablations

We perform an ablation study on our various proposed losses proposed, summarized in Tab. 4.3. We report average PCK@0.10 results for first 3 categories of the CUB-200 dataset and all 4 categories of the PF-Willow dataset. As expected, the keypoint loss $\mathcal{L}_{\text{KP}}$ plays the most important role in our overall framework. We also found the total variation regularization $\mathcal{L}_{\text{TV}}$ to be crucial for network training convergence. $\mathcal{L}_{\text{Equi}}$ is necessary for learning dense correspondence. Finally, $\mathcal{L}_{\text{Recon}}$ and $\mathcal{L}_{\text{Parts}}$ provide comparatively small improvements.

Table 4.3: **Ablation Study.** Average PCK@0.10 (for 3 and 4 categories respectively) in CUB and PF-Willow datasets.

Ablation	CUB-200 (3 categories)	PF-Willow (4 categories)
Complete objective	75.9	76.3
No $\mathcal{L}_{\text{KP}}$	22.8	36.2
No $\mathcal{L}_{\text{TV}}$	43.9	40.4
No $\mathcal{L}_{\text{Equi}}$	64.8	65.6
No $\mathcal{L}_{\text{Recon}}$	73.3	74.2
No $\mathcal{L}_{\text{Parts}}$	73.6	73.5

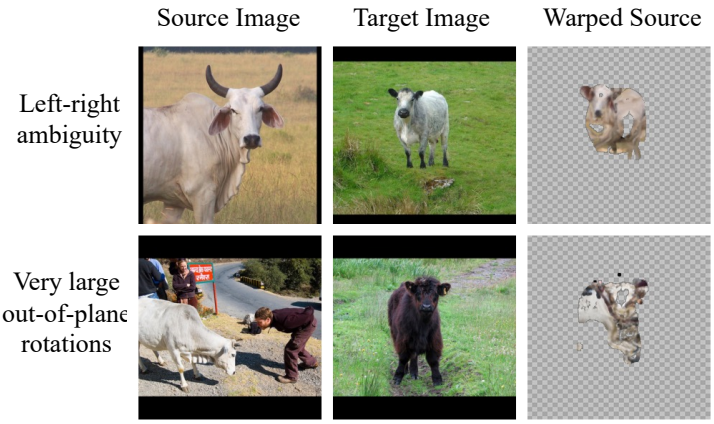


Figure 4.6: **Limitations.** Top row shows that our model can map left part of object in source to right part of object in target when object is symmetric. Bottom row shows that our model fails for very large out-of-plane rotations.

4.5 Limitations

Left-right ambiguity: One shortcoming of our approach is that it cannot differentiate well between left and right parts well for symmetric objects. We attribute this problem to the SSL models being invariant to left-right flips during their training. The top row of Fig. 4.6 shows that our model matches the left part of the cow torso in the source image to the right part of the torso in the target image (note left part of cow is not visible in the target image). Some heuristics used in prior works, such as flipping the source and target images and picking a combination that provides minimum total variation loss, could be used in our work as well. For brevity, we provide results without this heuristic.

Large shape changes: Our model doesn't handle large viewpoint changes well, especially when there are few intermediate viewpoints. In the bottom row of Fig. 4.6, we see that model is unable to warp the source cow image to target image even for the co-visible portions.

Part II: Learning to Compose Primitives

Chapter 5: Improved Modeling of 3D Shapes with Multi-view Depth Maps

Humans excel at perceiving the 3D structure of a large variety of objects in very diverse conditions. A combination of cues from signals such as color, texture, memory, and, most importantly, disparity from multi-view images enables 3D perception in humans. Biological evidence [198] suggests that the ventral cortex in our brain stores representations or features of various object configurations used while performing various tasks related to visual perception. We can naturally disentangle the identity of the object, and apply it on top of prior learned or memorized shape of objects. Some of the recent advances in neural network architectures primarily based on StyleGAN [199] generator [200, 201, 202, 203] attempt to accomplish the same by learning an initial embedding of say, faces, and adding noise vector to it in subsequent convolution layers to generate a new face. Improved neural network architectures along with large scale datasets and compute have helped us rapidly advance in the domain of representing and synthesizing images. However, due to the curse of dimensionality, we are still far from achieving such progress with 3D shapes, at scale. Two fundamental problems prevent us from bridging the gap between 2D and 3D synthesis - (1) finding an appropriate 3D representation and, (2) the availability of large-scale 3D datasets. In this paper, we postulate multi-view depth maps as a viable 3D representation for efficiently storing and generating high-quality 3D shapes. We show that architectures developed to work with 2D images can be easily adapted to disentangle viewpoint from shape or identity

information for 3D objects.

The use of multi-view representation for 3D objects has existed for a long time [204], however, it has been overshadowed in favor of representations such as meshes [205, 206, 207], point clouds [2, 208, 209], voxel grids [210, 211, 212], and more recently, implicit functions [1, 213, 214, 215, 216]. Deep learning methods, specifically CNNs, have shown excellent capabilities for modeling complex data distributions, such as images. Adopting CNNs to express continuous 3D surfaces with complex topologies only seems natural. However, non-image based 3D representations limit the application of CNNs due to several reasons. For example, voxel-based representations such as dense occupancy grid [210] grow cubically in memory and processing requirements and are hard to scale to achieve higher resolutions. Both point cloud-based and mesh-based representations can provide high-quality 3D shapes with less memory, however, learning on these kinds of representations is a challenging task since they require a new way of defining convolution and pooling operations. Meshes additionally can efficiently represent only fixed topology like faces and do not generalize to simple objects with varying topologies such as chairs and tables. Multi-view depth maps offer a promising alternative. They are memory efficient and can be directly used to store and visualize the 3D object from various viewpoints without explicitly storing the entire 3D object. Learning with CNNs on depth maps is relatively trivial. And last but not the least, with the availability of depth sensors on new generation smartphones, the amount of RGB-D data available for learning is only going to explode in the future.

In this work, we propose a framework for learning to reconstruct and synthesize 3D shapes using multi-view depth maps. We improve upon previously proposed multi-view approaches [217] in the following manners: (1) we show a simple way to adopt image-based CNN architectures to extract the identity of an object from one or more depth maps or silhouettes using an aver-

aging heuristic, and reconstruct the entire 3D shape from a single viewpoint, (2) our framework consumes very little memory and can learn from as few as two viewpoints of an object during training (3) using a single class conditional model, we show results competitive with methods that learn only using class-specific data in tasks such as single-view reconstruction.

5.1 Related Work

Learning good representations of 3D shapes from complex domains such as cars, chairs, humans, clothes, etc. is a classical problem in 3D Vision. Unlike 2D images, 3D shapes do not have a standard representation. Traditional approaches to modeling shape information have focused on identifying primitives that combine in a meaningful way to form existing shapes. [218] developed one of the early generative model for representing shapes as assembly of its parts. [219] proposed a component-based generative model that learns the probabilistic relationships between properties of shape components and relates them to learned underlying causes of structural variability (latent variables). [220] uses part-based templates to construct a probabilistic deformation model for generating shapes.

Recent efforts for 3D reconstruction and generation are primarily based on neural networks and can be broadly classified based on the 3D representation used: voxel-based, mesh-based, point-based, implicit function-based, or image-based methods. We discuss key recent works for each category below:

Voxel. Voxel-based representations allow for easily extending advances in 2D convolutions to 3D convolutions. Several works [221, 222] have attempted to learn the 3D shapes by reconstructing the voxel representation of the shape. 3DGAN [210] was the first GAN-based work

to synthesize 3D shapes by using a voxel grid to represent the shapes. Their work is a straight forward extension of DCGAN [223] to voxel volumes with the use of 3D convolutions. However, the use of 3D convolutions is very memory intensive and hard to scale to higher resolution voxel grids, making it difficult to represent shapes with fine details. Octree-based methods like [224], [212] make it possible to learn shapes up to a resolution of 512^3 by relieving the compute and memory limitations of dense voxel methods.

Mesh. Recent works [205, 225, 226] have exploited mesh representation of 3D shapes along with Graph Convolution Networks [227] for modeling human face and body. AtlasNet [228] and 3D-CODED [229] parameterize the surface of the 3D shape as a set of simple primitives and learn a mapping from the set of primitive shapes to the 3D surface. Such surface-parametric approaches rely on carefully chosen shape primitives and have shown promising results in reconstruction tasks.

Point Cloud. Point cloud is another way of representing 3D objects and closely matches the raw data from sensors like LiDARs. Early works [208, 209] show that distinctive 3D shapes can be learned using point cloud-based representation. [230] learns the shape embeddings of 3D objects using an auto-encoder framework and samples new objects by training a generative model on the learned embeddings. PointFlow [2] parameterizes 3D shapes as a two-level hierarchy of distributions, where the first level learns the distribution of shapes, and the second level learns the distribution of points within the shape using continuous normalizing flows. Though point cloud-based networks show promising results, the main limitation of learning with point clouds is that they lack topological information.

Implicit function. Recently, there has been an increased interest in using implicit functions to represent 3D shapes. Implicit functions assign a value to each 3D point. The set of points

representing a specific value represents the shape of the object. Occupancy Networks [231], IM-Net [213], and DeepSDF [1] learns the surface of an object represented as signed distance function (SDF) as a continuous decision boundary of a neural network. Implicit approaches are typically low memory and have shown to perform well at various tasks, however, the inference is usually very slow since each point in 3D still needs to be tested for presence or absence of the object.

Image. Image-based representations have demonstrated the potential for 3D shape understanding and reconstruction. Early works [232] show that the view-based descriptors learned using CNN can provide better representations for 3D shapes compared to the descriptors learned in 3D space. Owing to the efficient view-based representations of 3D shapes, several works have explored reconstructing 3D shapes from single or multi-view images. [233] attempted to generate images given the viewpoint and properties of the 3D shapes, albeit in a deterministic manner. [217] proposed a VAE-based approach to model 3D shapes using multi-view depth maps and silhouettes.

In Section 5.2, we discuss some of the limitations of the existing image-based methods and how we address them in our proposed framework. In Section 5.3, we evaluate our framework both quantitatively and qualitatively against approaches using different representations for 3D objects.

5.2 Our Approach

Existing neural network architectures for learning distributions over image data are oblivious to the 3D structure inscribed within images. A major challenge for the architectures working

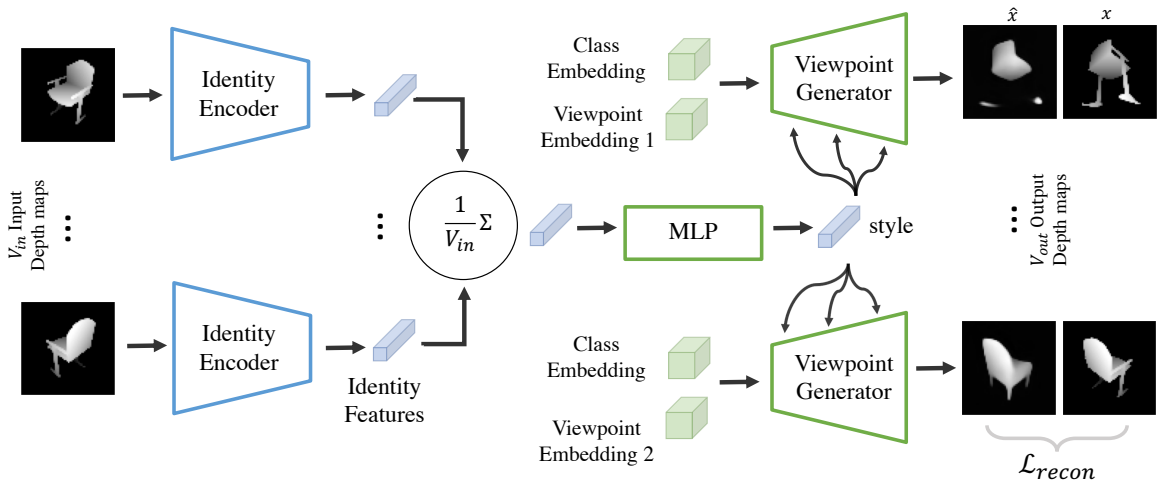


Figure 5.1: **Architecture**: Our Identity Encoder Network takes one or more depth maps of 3D object as input and encodes each of them into latent vectors. We use expected value of these latent vectors as the identity vector of the object. The decoder or viewpoint generator network uses this identity vector, category and viewpoint as input to generate depthmap of the object. It consists of an MLP that maps the identity vector to a style vector. This style vector is added to each block in the Viewpoint Generator Network using adaptive instance normalization.

with depth maps is to generate depth maps from multiple viewpoints that are consistent in the 3D space. Soltani et al. [217] resolve the multi-view consistency by treating depth maps and silhouettes from different viewpoints as various channels of the final output. Overall their model generates a $40 \times H \times W$ dimensional output representing 20 depth maps and 20 silhouettes, each of size $H \times W$. We hypothesize that representing a 3D shape as a 40 channel image poses two major challenges. First, a given spatial coordinate (pixel) in the 40 output channels does not correspond to the same location in 3D space, as the output channels correspond to different viewpoints. This is in stark contrast with typical three channel RGB or four channel RGBD images, where a pixel over all channels corresponds to a single point in 3D space (given by the intersection of camera ray with the scene). This also makes the training more challenging since each convolution filter aggregates features of a small neighborhood to predict features at a particular pixel location. However, a 40 channel pixel representing 20 locations in 3D would require

a much large neighborhood for accurate predictions. Second, having to predict all 40 channels simultaneously, makes it very memory intensive to work with large batch sizes. Smaller batch sizes often lead to unstable gradients which leads to the requirement of a small learning rate and longer training duration [234, 235]. For example, [217] uses a maximum batch size of 8 and a learning rate of 5×10^{-6}.

In this section, we discuss our approach for modeling 3D shapes using depth maps rendered from multiple viewpoints surrounding the object. We propose several improvements to overcome the shortcomings in existing depth maps-based approaches. We also show that the proposed model can trivially adapt to modern neural network architectures developed for learning image representation and generation.

5.2.1 Framework

Our overall framework is surprisingly simple, yet effective, and depicted in Figure 5.1. It consists of two components: (1) an identity encoder, that extracts viewpoint independent, shape specific representation of a 3D object, and (2) a viewpoint conditioned decoder, that generates the depth map corresponding to the encoded shape as seen from the given viewpoint. Unlike non-image based representations for 3D shapes, depth maps allows us to plugin and benefit from various neural network techniques developed for image-like data, such as convolutions, self-attention, or normalization layers. We hope that the simplicity of our method will bring more focus on image-based representations of 3D scenes and objects, like 2.5D, in the community. Next, we discuss each major component in detail.

Discrete Depth Maps. To keep our framework compatible with literature, we render depth maps

and silhouettes from 20 fixed camera viewpoints. As opposed to representing the depth map and silhouette as a two-channel image, we apply an 8-bit uniform quantization to the depth map. This expresses the depth in discrete values between 0 and 255, with 0 representing background and 1-255 representing the foreground. Though the maximum precision for depth values is reduced, the quality of shapes is significantly improved compared to continuous values. Similar approach of discretizing continuous signals has shown to be effective in [7, 236, 237]. Please refer to Appendix for additional details.

Identity Encoder. Our encoder takes as input a single depth map $\mathrm{R}^{C\times H\times W}$ and computes a view independent L-dimensional embedding of the depth map. During training, we obtain the disentangled shape identity from the viewpoint information using a simple averaging heuristic. We take two or more randomly sampled viewpoints of the same object and compute their embedding. We then take the expected value of this embedding and pass it on to the decoder. Since the decoding task can require the network to generate completely different viewpoints, the encoder is forced to learn identity information of the shape. The encoder starts with $H \times W$ resolution and applies a series of encoder blocks, each block reducing the resolution by half, recursively till we reach 4×4 resolution. Each encoder block consists of 3 convolutional layers and a downsampling layer. In our experiments, we also tried a viewpoint conditional variation of the encoder. More details of our identity encoder architecture can be found in Appendix.

Viewpoint Generator. Our framework comprises of a viewpoint generator that takes shape identity code generated by the identity encoder and a viewpoint to generate depth map of the shape as viewed from that viewpoint. The architecture of our generator is similar to the one in StyleGAN v2 [199, 200]. Instead of starting with latent, the generator starts with a constant viewpoint

embedding in shape of $256 \times 4 \times 4$ feature maps. The averaged latent vector obtained from the identity encoder is passed through a Multi-layered Perceptron (MLP) to generate a style vector, which is added to each layer of the viewpoint generator using AdaIN (Adaptive Instance Normalization) layers. For more details on the generator architecture, please refer to Figure 5.1 and the appendix.

Conditional generation. We extend our framework to be conditioned on the category of 3D objects. Along with the viewpoint embedding, we add an additional category embedding in shape of $256 \times 4 \times 4$ feature maps (same size as viewpoint embedding). This embedding is also learned during the training. Figure 5.5 shows a 2d visualization of this embedding plotted using TSNE [238].

5.2.2 Loss function and other training details

For each batch of B objects during training, we sample V_{in} depth maps per object. The depth maps are passed through the encoder to generate viewpoint invariant latent embeddings for each of the B objects. The generator then generates V_{out} number of depth maps for randomly selected viewpoints. We keep $V_{in} = V_{out} \leq V_{max} = 20$ constant throughout the training process. In ablation studies, we experiment with different values of 1, 2, 3 and 4 for V_{in} and V_{out}. Since our depth maps are discrete, at the end of the generator, we apply a softmax to get a probability distribution over 256 possible depth values. Instead of using a standard cross-entropy loss, we use a label smoothing loss [239]. For each ground truth depth value, we create a pseudo-ground truth distribution with a weight of $\epsilon = 0.2$ distributed uniformly at all locations except for the ground truth index which has a weight of $1 - \epsilon = 0.8$. We, finally minimize KL-Divergence loss

between the softmax predictions $\hat{x}$ and label-smoothed ground truth x, $\mathcal{L}(x, \hat{x}) = \mathbb{KL}[x \parallel \hat{x}]$. We use Adam optimizer [240] with a learning rate of 0.004, and a batch size of 32, in all our experiments.

5.2.3 Generative modeling with Implicit MLE

Given the trained identity encoder discussed above, we further extend our method to generate new 3D shapes by learning a latent space over the shape priors from the previous step using Implicit Maximum Likelihood Estimation (IMLE) [241]. We use IMLE since it addresses the three main challenges of training a Generative Adversarial Network – mode collapse, vanishing gradients, and training instability. It also claims to generate superior quality images compared to GANs and VAEs when trained for images [242].

Much similar to GAN, an IMLE uses an implicit model T to transform a noise vector e to the data distribution. Instead of learning from a discriminator, the parameters of the implicit model are learned by ensuring that in a random batch of training data samples, every sample is close to at least one of the transformed noise vectors. Empirically, we observe that training an IMLE on the depth maps directly results in blurry depth maps samples. So instead, we train a simple fully connected IMLE model T on latent vectors obtained from our identity encoder.

Our IMLE adaptation works as follows. Given the shape identities, at the start of each epoch, we first sample M noise vectors $\{e_1, e_2, \ldots, e_M\}$. The value of M is chosen to be twice the size of training data. A simple 2 hidden-layer fully connected feed-forward network T maps the noise vectors to $T(e_j)$ which has the same dimensions as shape identity vectors. Then, we sample a mini-batch of K training points. For each shape identity s_i in this mini-batch, nearest-

neighbour search finds the closest transformed noise vector $T(e_i)$.

$$e_i = \arg\min_{e_j} \|T(e_j) - s_i\|_2^2$$

The weights of network T are then learned by minimizing the distance between the nearest-neighbor correspondences

$$T = \arg\min_{\hat{T}} \sum_{i=1}^{K} \|s_i - \hat{T}(e_i)\|_2^2$$

Once the model is trained, we can simply sample a vector e from normal distribution, transform it to map to the distribution of shape vectors with a learned transformation $T(e)$. Finally, the viewpoint generator can then generate depth maps of this newly generated identity vector as viewed from different viewpoints, hence generating a new 3D shape.

5.3 Experiments

We evaluate our model both quantitatively and qualitatively on various tasks to compare it against the state-of-the-art approaches. We consider approaches that use various 3D representations, such as point clouds, voxel grids, implicit functions, and depth maps. We discuss four experiments to test the ability of our framework to learn the distribution of 3D data: (1) auto-encoding reconstructing seen and unseen shapes, (2) single view 3D reconstruction, (3) analyze the smoothness of the embeddings, and (4) sample new shapes using the learned model. Since most approaches train an independent model for each category, we also train independent unconditional models for 6 of the categories of ShapeNet (airplane, car, chair, lamp, sofa, and table), for quantitative comparisons. However, for qualitative analyses, we use the class conditional model.

5.3.1 Dataset

We use ShapeNetCore v2 dataset [211, 222] which consists of 52472 3D aligned models from 55 categories for all our experiments. We further use the provided train, validation, and test splits of 36814, 5306 and 10276 models. To render depth maps, we follow the same procedure as [217] and place 20 virtual cameras at 20 vertices of a regular dodecahedron enclosing the object. All cameras are assumed to be located at a fixed distance of 2.5m from the origin and point towards the origin. The focal length of the camera used is 50mm and field of view is 40°. We use Blender [243], an open source 3D model creation and rendering suite to render the depth maps and silhouettes.

5.3.2 Baselines

We select a variety of representative baseline approaches for modeling 3D shapes.

3D-EPN. [244] proposed a voxel-based approach for shape completion of 3D shapes. Their model consists of a sequence of 3D convolution layers to predict missing voxels.

AtlasNet. [228] parameterizes meshes as surface elements, and tries to auto-encode the mesh or infer the mesh using a single view of the object.

DeepSDF. [1] learns continuous Signed Distance Function (SDF) where the zero-level-set of the learned function represents the shape's surface. A shape embedding, with a Gaussian prior, is learned in an auto-decoder setting. At inference time, the latent shape code is estimated using MAP from the full or partial observations of 3D shape.

Soltani et al. [217] is perhaps closest to our work since they also work with multi-view depth maps and follow a similar encoder-decoder approach.

Table 5.1: Overview of baselines

Method	Representation	Model size (GB)	Inference time (s)
3D-EPN [244]	Voxel	0.42	-
AtlasNet-25 [228]	Mesh	0.17	0.32
DeepSDF [1]	SDF	0.01	9.72
Soltani et al. [217]	Multiview Depth	0.39	0.03
PointFlow [2]	Point cloud	**0.005**	18.87
Ours	Multiview Depth	0.41	**0.01**

PointFlow. [2] takes a probabilistic approach and learns a two-level distribution to model 3D data, each learned using continuous normalizing flow. The first level learns the distribution of shapes and second learns the distribution of points for a given shape.

5.3.3 Metrics

We use the following to compare a pair of point clouds:

Chamfer Distance (CD): is the sum of distances between points from a set to its nearest neighbors in the other set.

Earth Mover's Distance (EMD): is used for measuring optimal transport distance between two discrete distributions.

Both CD and EMD don't work well if large holes exist in either source or target models. For CD, we use 30000 points and compute the distance in both directions in order to make it symmetric. The reported CD is multiplied by 10^3. EMD is slower to compute and we use a small sample of 500 points from each point cloud in all our evaluations.

We follow the protocols of [1] to evaluate our model's capability in several ways. To compare against other approaches quantitatively, we project the depth maps generated by model

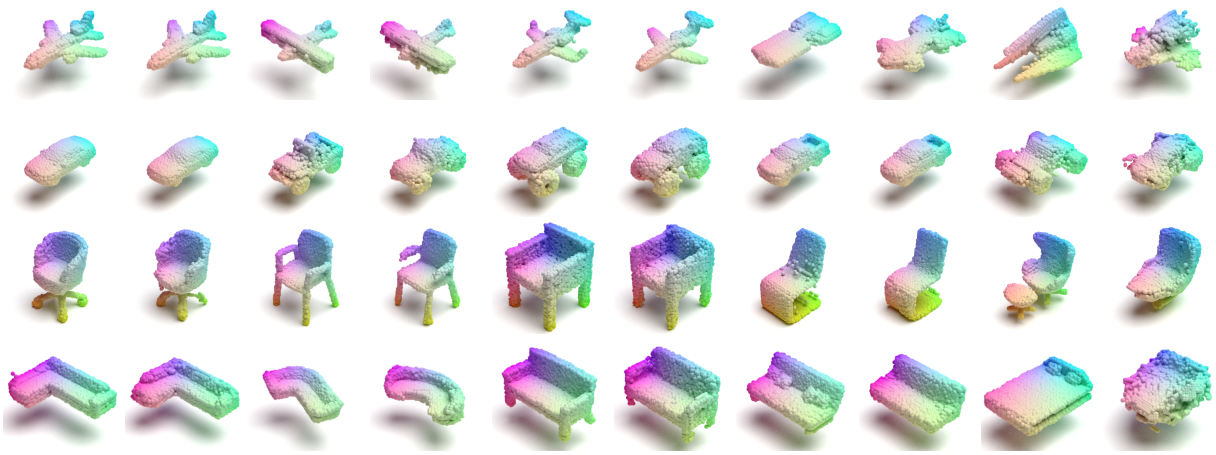


Figure 5.2: **Reconstruction on test objects.** We evaluate our model's ability to encode geometry of 3D shapes with its latent representation. Each row represents 3D shapes from different categories. Odd columns are the input objects and even columns are the reconstructions. We render depth maps of input objects from different viewpoints. Our encoder encodes each of the depth map independently and outputs a latent code. We average the latent code over all the viewpoints of the object. Finally we use our generator to use the same latent code and generate multiple viewpoints from the latent code. Final output depth maps are projected back to 3D. Last two columns shows some of the failure cases likely because of lack of similar samples in training data.

to 3D point clouds. Specifically, we measure (1) how well our model can faithfully reconstruct unseen 3D shapes with a single latent vector while preserving the geometric details, and (2) if our model can extrapolate complete 3D structure from a single viewpoint information. We also analyze (1) the latent space learned by the model both in terms of smoothness, and (2) semantics of the category embeddings learned by the model. Finally we show the generative capabilities of the model, using an IMLE approach as discussed in Section 5.2.3.

5.3.4 Results

Auto-encoding. We first evaluate the model's ability to represent a given 3D shape with a fixed latent vector size. We follow [1] and compute mean/median CD, and mean EMD between ground truth point clouds and point clouds reconstructed from depth maps. Note that our model optimization is done in 2D depth map space and not for either of these metrics but we are able to

Table 5.2: Reconstruction on test data. We measure the reconstruction performance of different techniques on the test dataset. [1] and [2] outperforms the reconstruction in majority of the cases.

CD, mean	chair	sofa	table	lamp	plane	car
AtlasNet-Sph [228]	0.75	0.45	0.73	2.38	0.19	-
AtlanNet-25 [228]	0.37	0.41	**0.33**	1.18	0.22	-
DeepSDF [1]	**0.20**	**0.13**	0.55	**0.83**	0.14	-
Soltani et al. [217]	1.32	0.88	-	3.20	1.82	-
PointFlow [2]	0.75	-	-	-	**0.07**	0.40
Ours	0.69	0.57	1.33	2.19	0.44	**0.36**
CD, median	chair	sofa	table	lamp	plane	car
AtlasNet-Sph[228]	0.51	0.33	0.39	2.18	0.08	-
AtlanNet-25[228]	0.28	0.31	0.20	0.99	0.07	-
DeepSDF [1]	**0.07**	**0.09**	**0.07**	**0.22**	**0.04**	-
Soltani et al. [217]	1.28	0.76	-	1.99	1.71	-
PointFlow [2]	0.55	-	-	-	0.05	0.36
Ours.	0.34	0.28	0.38	0.89	0.23	**0.19**
EMD, mean	chair	sofa	table	lamp	plane	car
AtlasNet-Sph[228]	0.071	0.050	0.060	0.085	0.038	-
AtlanNet-25[228]	0.064	0.063	0.073	0.062	0.041	-
DeepSDF [1]	**0.049**	**0.047**	0.050	**0.059**	**0.033**	-
Soltani et al. [217]	0.139	0.072	0.075	0.096	0.063	-
PointFlow [2]	0.078	-	-	-	0.039	0.066
Ours	0.077	0.078	**0.049**	0.099	0.056	**0.046**

perform competitively against voxel and mesh-based methods. Table 5.2 shows point cloud and SDF based approaches perform better at these reconstruction metrics. The trade-off is reduced performance in single view reconstruction as discussed next. Figure 5.2 shows the reconstruction results on the test set.

Single View 3D Reconstruction (SVR). Next, we consider the problem of recovering the 3D object from a given depth map from a single viewpoint. Our model can trivially perform this task since encoder encodes view independent representation of the object and decoder can use the encoded latent vector to generate depth maps from any viewpoint. For quantitative evaluation, we take the single viewpoint depth map of objects from test split and compute CD and EMD loss

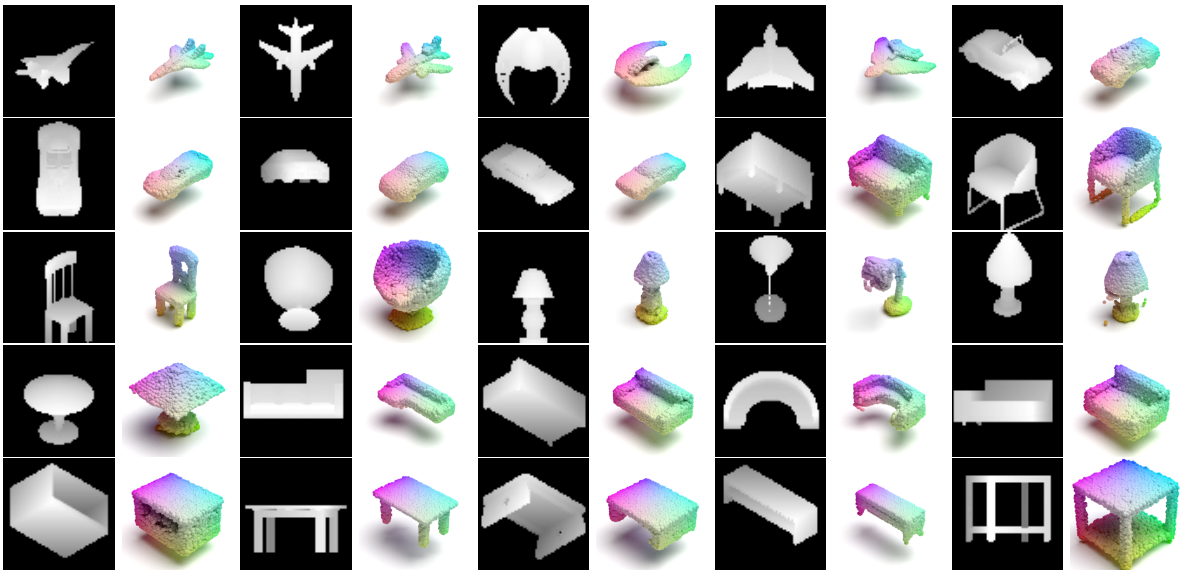


Figure 5.3: **Single View Reconstruction.** Given a single input depth map, we show reconstructions of complete 3D object using our approach. Odd columns are the input depth maps, even columns are the full 3D reconstructions viewed from a canonical viewpoint.

Table 5.3: Shape reconstruction from a single viewpoint

CD, mean	chair	sofa	table	lamp	plane	car
3D-EPN [244]	2.83	2.18	-	-	2.19	-
DeepSDF [1]	2.11	1.59	-	-	1.16	-
Soltani et al. [217].	1.89	3.07	**1.99**	7.18	2.71	-
Ours	**1.05**	**0.85**	2.26	**2.72**	**0.63**	**0.59**
CD, median	chair	sofa	table	lamp	plane	car
3D-EPN [244]	2.25	2.03	-	-	1.63	-
DeepSDF [1]	1.28	0.82	-	-	0.37	-
Soltani et al. [217]	1.58	1.29	1.41	3.16	2.27	-
Ours	**0.49**	**0.42**	**0.62**	**1.00**	**0.26**	**0.25**
EMD, mean	chair	sofa	table	lamp	plane	car
3D-EPN [244]	0.084	0.071	-	-	0.063	-
DeepSDF [1]	**0.071**	**0.059**	-	-	**0.049**	-
Soltani et al. [217]	0.150	0.091	0.086	0.143	0.083	-
Ours	0.091	0.090	**0.067**	**0.109**	0.064	**0.056**

between the full ground truth and predicted 3D point clouds. Table 5.3 shows that our method outperforms other methods in SVR with CD as distance metric. Our method is close second (and better than other multi-view depth baseline) when we use EMD. Figure 5.3 shows the results of

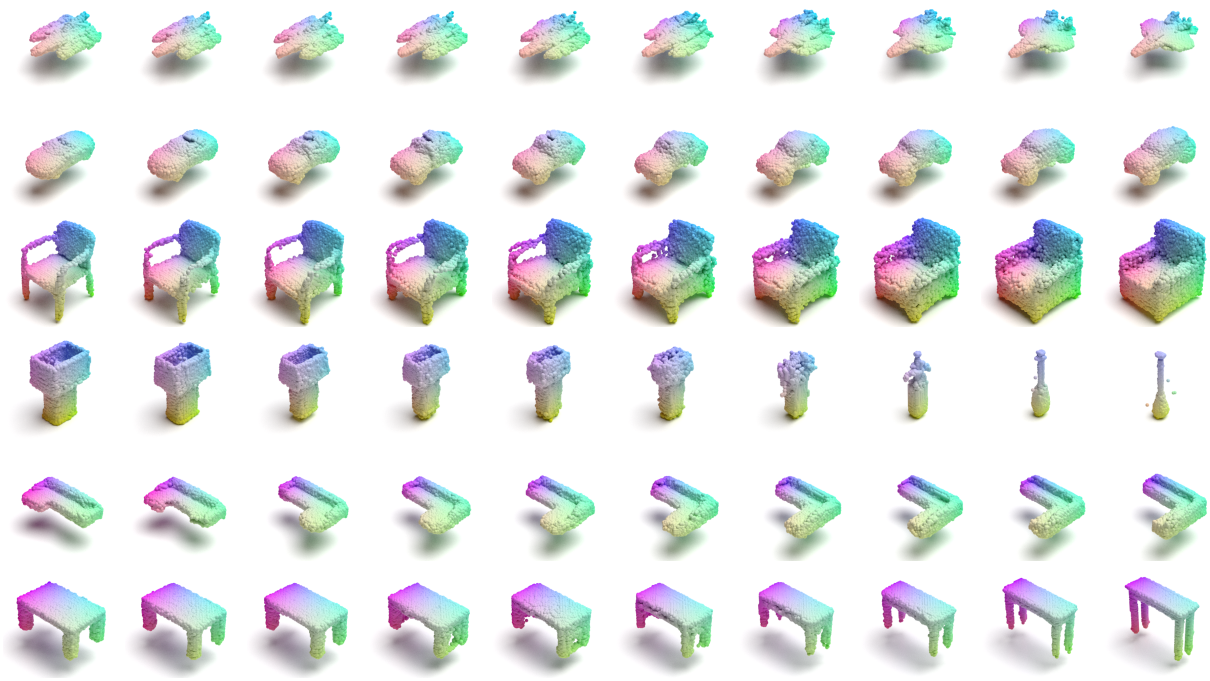


Figure 5.4: **Shape Interpolation.** The latent representations learned by our model are smooth. In this figure, we interpolate between two test objects (left-most and right-most column) by generating the intermediate 3D objects from the spherical interpolation of the latent codes.

SVR using our model.

3D shape interpolation. We choose spherical over linear interpolation for analyzing the latent space as recommended by [200]. Figure 5.4 shows samples generated by interpolating between two random objects picked from the test data.

Learned class embeddings Our class conditional model learns embeddings for each of the class. Figure 5.5 shows a TSNE [238] plot of the embeddings in two dimensions. We can immediately observe that object classes that look visually similar appear closer in the embedding space.

3D Shape Synthesis. We train an IMLE-based model on the shape identities obtained from the trained identity encoder. Once the model is trained, we use it to sample the shape identities of new 3D objects. The viewpoint generator maps the shape identities to the depth map of the object as viewed from the given viewpoint. Figure 5.6 shows a few generated 3D objects for

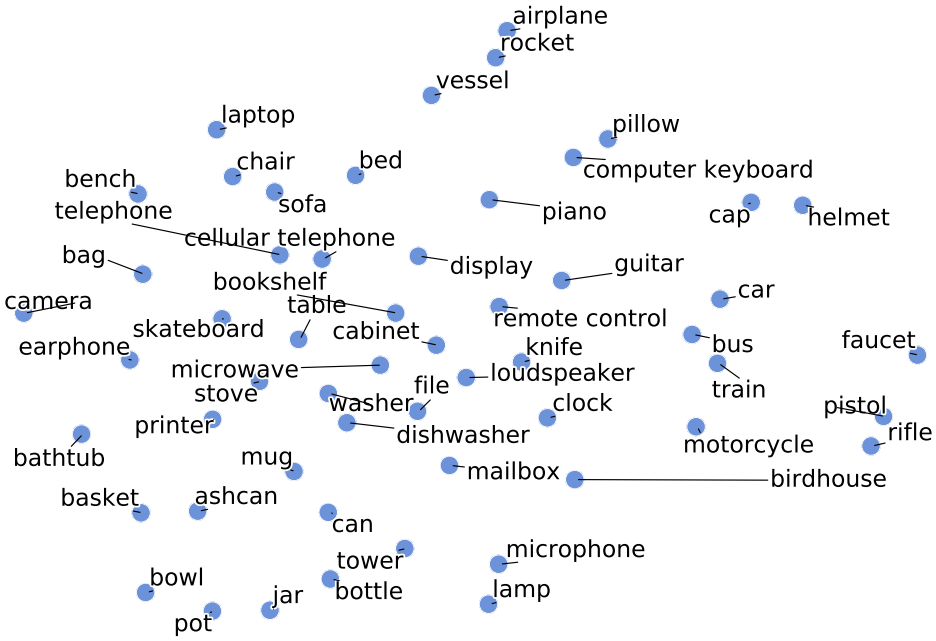


Figure 5.5: Word embeddings learned by our model for various class objects. We see that visually similar categories cluster together, even if they are not semantically similar. E.g., airplane, rocket and vessel are together, mailbox and microwave are also together.

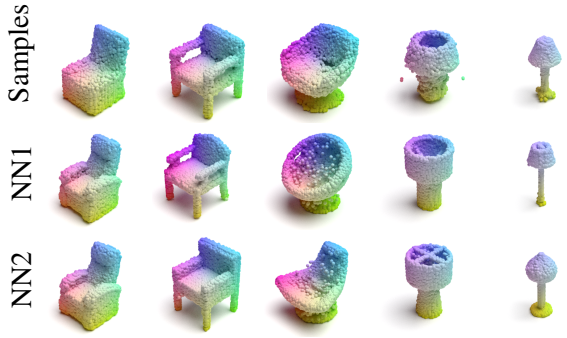


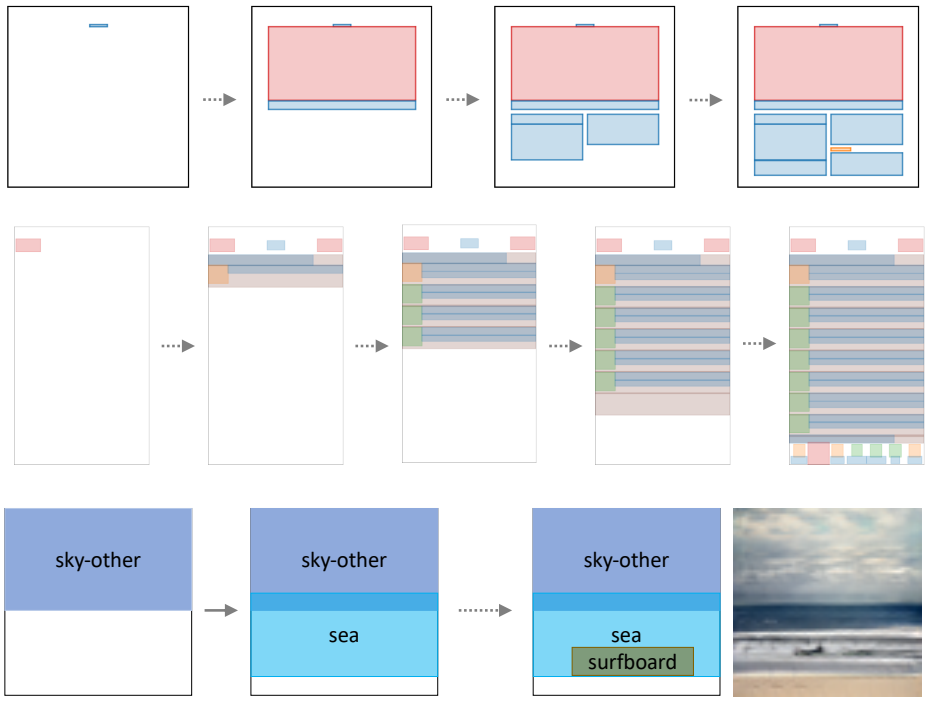
Figure 5.6: Two NN to generated 3D models: our model produces 3D models that differ from the two closest train samples.

two categories, along with the nearest neighbors for each generated sample. We observe that the generated samples are diverse and differ from their closest neighbors in the training set.

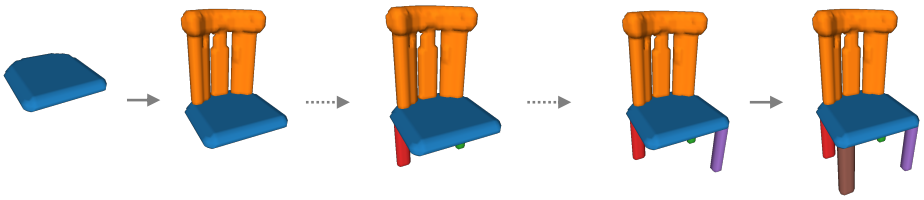
Chapter 6: Layout Generation and Completion with Self-attention

In the real world, there exists a strong relationship between different objects that are found in the same environment [245, 246]. For example, a dining table usually has chairs around it, a surfboard is found near the sea, horses do not ride cars *etc*. [247] provided strong evidence in cognitive neuroscience that perceiving and understanding a scene involves two related processes: *perception* and *comprehension*. Perception deals with processing the visual signal or the appearance of a scene. Comprehension deals with understanding the *schema* of a scene, where this schema (or layout) can be characterized by contextual relationships (*e.g.*, support, occlusion, and relative likelihood, position, and size) between objects. For generative models that synthesize scenes, this evidence underpins the importance of two factors that contribute to the *realism* or plausibility of a generated scene: layout, *i.e.*, arrangement of different objects, and their appearance (in terms of pixels). Generating a realistic scene necessitates both the factors to be plausible.

The advancements in the generative models for image synthesis have primarily targeted plausibility of the appearance signal by generating incredibly realistic images often with a single entity such as faces [248, 249], or animals [250, 251]. In the case of large and complex scenes, with strong non-local relationships between different elements, most methods require proxy representations for layouts to be provided as inputs (*e.g.*, scene graph, segmentation mask, sentence).



(a) Autoregressive 2D layout generation and downstream Layout-to-Image application



(b) Generating 3D objects autoregressively

Figure 6.1: Our framework can synthesize layouts in diverse natural as well as human designed data domains such as documents, mobile app wireframes, natural scenes or 3D objects in a sequential manner.

We argue that to plausibly generate large scenes without such proxies, it is necessary to understand and generate the layout of a scene, in terms of contextual relationships between various objects present in the scene.

Learning to generate layouts is useful for several stand-alone applications that require generating layouts or templates with/without user interaction. For instance, in the UI design of mobile apps and websites, an automated model for generating plausible layouts can significantly decrease the manual effort and cost of building such apps and websites. Finally, a model to cre-

ate layouts can potentially help generate synthetic data for various tasks tasks [252, 253, 254, 255, 256]. Figure 6.1 shows some of the layouts autoregressively generated by our approach in various domains such as documents, mobile apps, natural scenes, and 3D shapes.

Formally, a scene layout can be represented as an unordered set of graphical primitives. The primitive itself can be discrete or continuous depending on the data domain. For example, in the case of layout of documents, primitives can be bounding boxes from discrete classes such as 'text', 'image', or 'caption', and in case of 3D objects, primitives can be 3D occupancy grids of parts of the object such as 'arm', 'leg', or 'back' in case of chairs. Additionally, in order to make the primitives compositional, we represent each primitive by a location vector with respect to the origin, and a scale vector that defines the bounding box enclosing the primitive. Again, based on the domain, these location and scale vectors can be 2D or 3D. A generative model for layouts should be able to look at all existing primitives and propose the placement and attributes of a new one. We propose a novel framework LayoutTransformer that first maps the different parameters of the primitive independently to a fixed-length continuous latent vector, followed by a masked Transformer decoder to look at representations of existing primitives in layout and predict the next primitive (one parameter at a time). Our generative framework can start from an empty set, or a set of primitives, and can iteratively generate a new primitive one parameter at a time. Moreover, by predicting either to stop or to generate the next primitive, our approach can generate variable length layouts. Our **main contributions** can be summarized as follows:

- We propose LayoutTransformer a simple yet powerful auto-regressive model that can synthesize new layouts, complete partial layouts, and compute likelihood of existing layouts. Self-attention approach allows us to visualize what existing elements are important for

generating the next category in the sequence.

- We model different attributes of layout elements separately - doing so allows the attention module to more easily focus on the attributes that matter. This is important especially in datasets with inherent symmetries such as documents or apps and in contrast with existing approaches which concatenate or fuse different attributes of layout primitives.

- We present an exciting finding – encouraging a model to understand layouts results in feature representations that capture the semantic relationships between objects automatically (without explicitly using semantic embeddings, like word2vec [3]). This demonstrates the utility of the task of layout generation as a proxy-task for learning semantic representations,

- LayoutTransformer shows good performance with essentially the same architecture and hyperpameters across very diverse domains. We show the adaptability of our model on four layout datasets: MNIST Layout [257], Rico Mobile App Wireframes [258], PubLayNet Documents [259], and COCO Bounding Boxes [260]. To the best of our knowledge, MMA is the first framework to perform competitively with the state-of-the-art approaches in 4 diverse data domains.

6.1 Related Work

Generative models. Deep generative models based on CNNs such as variational auto-encoders (VAEs) [261], and generative adversarial networks (GANs) [34] have recently shown a great promise in terms of faithfully learning a given data distribution and sampling from it. There has also been research on generating data sequentially [236, 262] even when the data has no natural

order [263]. Many of these approaches often rely on low-level information [12] such as pixels while generating images [248, 250], videos [264], or 3D objects [1, 2, 265, 266] and not on semantic and geometric structure in the data.

Scene generation. Generating 2D or 3D scenes conditioned on sentence [267, 268, 269], a scene graph [6, 270, 271], a layout [272, 273, 274, 275] or an existing image [276] has drawn a great interest in vision community. Given the input, some works generate a fixed layout and diverse scenes [5], while other works generate diverse layouts and scenes [6, 267]. These methods involve pipelines often trained and evaluated end-to-end, and surprisingly little work has been done to evaluate the layout generation component itself. Layout generation serves as a complementary task to these works and can be combined with these methods. In this work, we evaluate the layout modeling capabilities of two of the recent works [6, 267] that have layout generation as an intermediate step. We also demonstrate the results of our model with Layout2Im [5] for image generation.

Layout generation. The automatic generation of layouts is an important problem in graphic design. Many of the recent data-driven approaches use data specific constraints in order to model the layouts. For example, [277, 278, 279, 280] generates top-down view indoor rooms layouts but make several assumptions regarding the presence of walls, roof *etc*., and cannot be easily extended to other datasets. In this paper, we focus on approaches that have fewer domain-specific constraints. LayoutGAN [257] uses a GAN framework to generate semantic and geometric properties of a fixed number of scene elements. LayoutVAE [4] starts with a label set, *i.e*., categories of all the elements present in the layout, and then generates a feasible layout of the scene. [281] attempt to generate document layouts given the images, keywords, and category of the document. [282] proposes a method to construct hierarchies of document layouts using a recursive

variational autoencoder and sample new hierarchies to generate new document layouts. [283] develops an auto-encoding framework for layouts using Graph Networks. 3D-PRNN [284], PQ-Net [285] and ComplementMe [286], generates 3D shapes via sequential part assembly. While 3D-PRNN generates only bounding boxes, PQ-Net and ComplementMe can synthesize complete 3D shapes starting with a partial or no input shape.

Our approach offers several advantages over current layout generation approaches without sacrificing their benefits. By factorizing primitives into structural parameters and compositional geometric parameters, we can generate high-resolution primitives using distributed representations and consequently, complete scenes. The autoregressive nature of the model allows us to generate layouts of arbitrary lengths as well as start with partial layouts. Further, modeling the position and size of primitives as discrete values (as discussed in §6.2.1) helps us realize better performance on datasets, such as documents and app wireframes, where bounding boxes of layouts are typically axis-aligned. We evaluate our method both quantitatively and qualitatively with state-of-the-art methods specific to each dataset and show competitive results in very diverse domains.

6.2 Our Approach

In this section, we introduce our attention network in the context of the layout generation problem. We first discuss our representation of layouts for primitives belonging to different domains. Next, we discuss the LayoutTransformer framework and show how we can leverage Transformers [105] to model the probability distribution of layouts. MMA allows us to learn non-local semantic relationships between layout primitives and also gives us the flexibility to

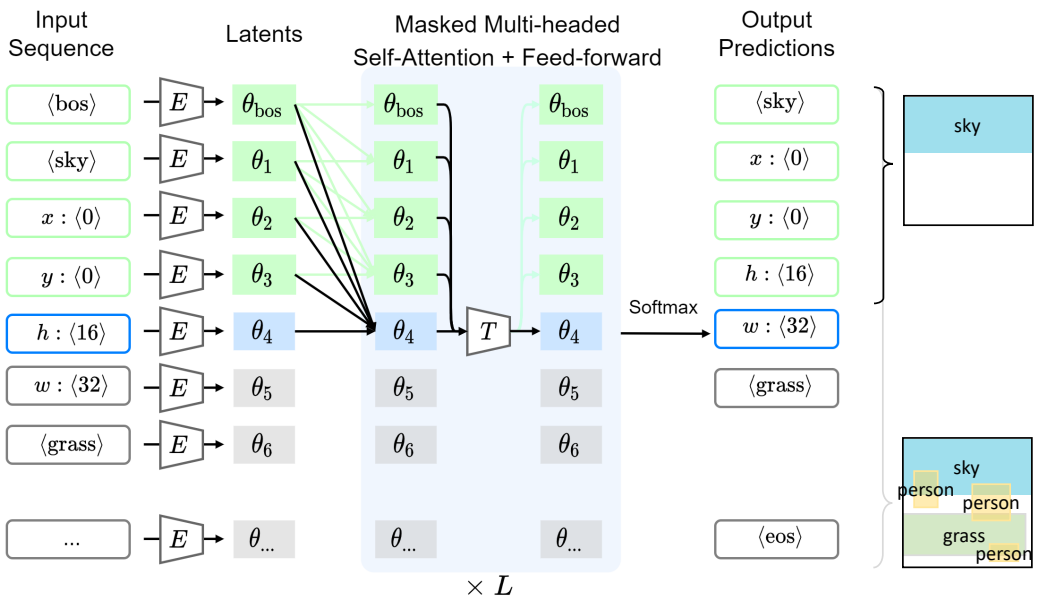


Figure 6.2: The architecture depicted for a toy example. LayoutTransformer takes layout elements as input and predicts the next layout elements as output. During training, we use teacher forcing, *i.e.*, use the ground-truth layout tokens as input to a multi-head decoder block. The first layer of this block is a masked self-attention layer, which allows the model to see only the previous elements in order to predict the current element. We pad each layout with a special ⟨bos⟩ token in the beginning and ⟨eos⟩ token in the end.

work with variable length layouts.

6.2.1 Layout Representation

Given a dataset of layouts, a single layout instance can be defined as a graph $\mathcal{G}$ with n nodes, where each node $i \in \{1, \ldots, n\}$ is a graphical primitive. We assume that the graph is fully-connected, and let the attention network learn the relationship between nodes. The nodes can have structural or semantic information associated with them. For each node, we project the information associated with it to a d-dimensional space represented by feature vector $\mathbf{s}_i$. Note that the information itself can be discrete (*e.g.*, part category), continuous (*e.g.*, color), or multi-dimensional vectors (*e.g.*, signed distance function of the part) on some manifold. Specifically,

in our ShapeNet experiments, we use an MLP to project part embedding to d-dimensional space, while in the 2D layout experiments, we use a learned d-dimensional category embedding which is equivalent to using an MLP with zero bias to project one-hot encoded category vectors to the latent space.

Each primitive also carries geometric information $\mathbf{g}_i$ which we factorize into a position vector and a scale vector. For the layouts in $\mathbb{R}^2$ such as images or documents, $\mathbf{g}_i = [x_i, y_i, h_i, w_i]$, where (x, y) are the coordinates of the centroid of primitive and (h, w) are the height and width of the bounding box containing the primitive, normalized with respect to the dimensions of the entire layout.

Representing geometry with discrete variables. We apply an 8-bit uniform quantization on each of the geometric fields and model them using Categorical distribution. Discretizing continuous signals is a practice adopted in previous works for image generation such as PixelCNN++ [287], however, to the best of our knowledge, it has been unexplored in the layout modeling task. We observe that even though discretizing coordinates introduces approximation errors, it allows us to express arbitrary distributions which we find particularly important for layouts with strong symmetries such as documents and app wireframes. We project each geometric field of the primitive independently to the same d-dimension, such that i^{th} primitive in $\mathbb{R}^2$ can be represented as $(\mathbf{s}_i, \mathbf{x}_i, \mathbf{y}_i, \mathbf{h}_i, \mathbf{w}_i)$. We concatenate all the elements in a flattened sequence of their parameters. We also append embeddings of two additional parameters $\mathbf{s}_{\langle\text{bos}\rangle}$ and $\mathbf{s}_{\langle\text{eos}\rangle}$ to denote start & end of sequence. Layout in $\mathbb{R}^2$ can now be represented by a sequence of $5n + 2$ latent vectors.

$$\mathcal{G} = (\mathbf{s}_{\langle\text{bos}\rangle}; \mathbf{s}_1; \mathbf{x}_1; \mathbf{y}_1; \mathbf{h}_1; \mathbf{w}_1; \ldots; \mathbf{s}_n; \mathbf{x}_n; \mathbf{y}_n; \mathbf{h}_n; \mathbf{w}_n; \mathbf{s}_{\langle\text{eos}\rangle})$$

For brevity, we use $\boldsymbol{\theta}_j$, $j \in \{1, \ldots, 5n+2\}$ to represent any element in the above sequence. We can now pose the problem of modeling this joint distribution as product over series of conditional distributions using chain rule:

$$p(\boldsymbol{\theta}_{1:5n+2}) = \prod_{j=1}^{5n+2} p(\boldsymbol{\theta}_j | \boldsymbol{\theta}_{1:j-1}) \tag{6.1}$$

Figure 6.3: **Generated 3D objects.** Top row shows input primitives to the model. Bottom row shows the layout obtained by our approach.

6.2.2 Model architecture and training

Our overall architecture is shown in Fig. 6.2. Given an initial set of K visible primitives (where K can be 0 when generating from scratch), our attention based model takes as input, a random permutation of the visible nodes, $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_K)$, and consequently a sequence of d-dimensional vectors $(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{5K})$. We find this to be an important step since by factorizing primitive representation into geometry and structure fields, our attention module can explicitly assign weights to individual coordinate dimensions. The attention module is similar to Transformer Decoder [105] & consists of L attention layers, each comprising of (a) a masked multi-head attention layer ($\mathbf{h}^{\text{attn}}$), and (b) fully connected feed forward layer ($\mathbf{h}^{\text{fc}}$). Each sublayer also adds

residual connections [55] and LayerNorm [288].

$$\hat{\boldsymbol{\theta}}_j = \text{LayerNorm}(\boldsymbol{\theta}_j^{l-1} + \mathbf{h}^{\text{attn}}(\boldsymbol{\theta}_1^{l-1}, \ldots, \boldsymbol{\theta}_{j-1}^{l-1})) \tag{6.2}$$

$$\boldsymbol{\theta}_j^{l} = \text{LayerNorm}(\hat{\boldsymbol{\theta}}_j + \mathbf{h}^{\text{fc}}(\hat{\boldsymbol{\theta}}_j)) \tag{6.3}$$

where l denotes the layer index. Masking is performed such that $\boldsymbol{\theta}$ only attends to all the input latent vectors as well as previous predicted latent vectors. The output at the last layer corresponds to next parameter. At training and validation time, we use teacher forcing, *i.e.*, instead of using output of previous step, we use groundtruth sequences to train our model efficiently.

Loss. We use a softmax layer to get probabilities if the next parameter is discrete. Instead of using a standard cross-entropy loss, we minimize KL-Divergence between softmax predictions and output one-hot distribution with Label Smoothing [239], which prevents the model from becoming overconfident. If the next parameter is continuous, we use an L^1 loss.

$$\begin{aligned}\mathcal{L} = \mathbb{E}_{\boldsymbol{\theta} \sim \text{Disc.}}[\, D_{\text{KL}}(\text{SoftMax}(\boldsymbol{\theta}^L) \parallel p(\boldsymbol{\theta}'))\,] \\ + \lambda \mathbb{E}_{\boldsymbol{\theta} \sim \text{Cont.}}[\, ||\boldsymbol{\theta} - \boldsymbol{\theta}'||_1\,]\end{aligned}$$

3D Primitive Auto-encoding. PartNet dataset [289] consists of 3D objects decomposed into simpler meaningful primitives, such as chairs are composed of back, arms, 4 legs, and so on. We pose the problem of 3D shape generation as generating a layout of such primitives. We use [290]'s approach to first encode voxel-based represent of primitive to d-dimensional latent space using 3D CNN. An MLP based implicit parameter decoder projects the latent vector to the surface occupancy grid of the primitive.

Order of primitives. One of the limitations of an autoregressive modeling approach is that sequence of primitives is an important consideration, in order to train the generative model, even if the layout doesn't have a natural defined order [263]. To generate a layout from any partial layout, we use a random permutation of primitives as input to the model. For the output, we always generate the sequences in raster order of centroid of primitives, *i.e.*, we order the primitives in ascending order of their (x, y, z) coordinates. In our experiments, we observed that the ordering of elements is important for model training. Note that similar limitations are faced by contemporary works in layout generation [4, 267, 277, 291], image generation [30, 287] and 3D shape generation [284, 285]. Generating a distribution over an order-invariant set of an arbitrary number of primitives is an exciting problem and we will explore it in future research.

Other details. In our base model, we use $d = 512$, $L = 6$, and $n_{\text{head}} = 8$ (number of multi-attention heads). Label smoothing uses an $\epsilon = 0.1$, and $\lambda = 1$. We use Adam optimizer [240] with $\beta_1 = 0.9, \beta_2 = 0.99$ and learning rate 10^{-4} (10^{-5} for PartNet). We use early stopping based on validation loss. In the ablation studies provided in appendix, we show that our model is quite robust to these choices, as well as other hyperparameters (layout resolution, ordering of elements, ordering of fields). To sample a new layout, we can start off with just a start of sequence embedding or an initial set of primitives. Several decoding strategies are possible to recursively generate primitives from the initial set. In samples generated for this work, unless otherwise specified, we have used nucleus sampling [292], with top-$p = 0.9$ which has been shown to perform better as compared to greedy sampling and beam search [293].

6.3 Experiments

In this section, we discuss the qualitative and quantitative performance of our model on different datasets. Evaluation of generative models is hard, and most quantitative measures fail in providing a good measure of novelty and realism of data sampled from a generative model. We will use dataset-specific quantitative metrics used by various baseline approaches and discuss their limitations wherever applicable. We will provide the code and pretrained models to reproduce the experiments.

6.3.1 3D Shape synthesis (on PartNet dataset)

PartNet is a large-scale dataset of common 3D shapes that are segmented into semantically meaningful parts. We use two of the largest categories of PartNet - Chairs and Lamp. We voxelize the shapes into 64^3 and train an autoencoder to learn part embeddings similar to the procedure followed by PQ-Net [285]. Overall, we had 6305 chairs and 1188 lamps in our datasets. We use the official train, validation, & test split from PartNet. Although it is fairly trivial to extend our method to train for the class-conditional generation of shapes, in order to compare with baselines fairly, we train separate models for each of the categories.

Generated Samples. Fig. 6.3 shows examples of shape completion from the PartNet dataset. Given a random primitive, we use our model to iteratively predict the latent shape encoding of the next part, as well its position and scale in 3D. We then use the part decoder to sample points on the surface of the object. For visualization, we use the marching cubes algorithm to generate a mesh and render the mesh using a fixed camera viewpoint.

Quantitative Evaluation. The output of our model is point clouds sampled on the surface of

Table 6.1: Evaluation of generated shapes in Chair category. The best numbers are in bold, second-best are underlined

Method	JSD↓	MMD(CD)↓	MMD(EMD)↓	Cov(CD)↑	Cov(EMD)↑	1-NNA(CD)↓	1-NNA(EMD)↓
PointFlow [2]	1.74	2.42	<u>7.87</u>	46.83	46.98	<u>60.88</u>	<u>59.89</u>
StructureNet [294]	4.77	0.97	15.24	29.67	31.7	75.32	74.22
IM-Net [290]	0.84	**0.74**	12.28	52.35	54.12	68.52	67.12
PQ-Net [285]	<u>0.83</u>	0.83	14.16	<u>54.91</u>	**60.72**	71.31	67.8
Ours	**0.81**	<u>0.79</u>	**7.38**	**55.25**	<u>55.44</u>	**60.67**	**59.11**

the 3D shapes. We use Chamfer Distance (CD) and Earth Mover's Distance (EMD) to compare two point clouds. Following prior work, we use 4 different metrics to compare the distribution of shapes generated from the model and shapes in the test dataset: (i) Jensen Shannon Divergence (JSD) computes the KL divergence between marginal distribution of point clouds in generated set and test set, (ii) Coverage (Cov) - compares the distance between each point in generated set to its nearest neighbor in test set, (iii) Minimum Matching Distance (MMD) - computes the average distance of each point in test set to its nearest neighbor in generated set, and (iv) 1-nearest neighbor accuracy (1-NNA) uses a 1-NN classifier see if the nearest neighbor of a generated sample is coming from generated set or test set. Our model performs competitively with existing approaches to generate point clouds. Table 6.1 shows the generative performance of our model in the 'Chair' category, with respect to recent proposed approaches. Our model's performance is either the best or second-best in all the metrics we evaluated in this work.

6.3.2 Layouts for natural scenes

COCO bounding boxes dataset is obtained using bounding box annotations in COCO Panoptic 2017 dataset [260]. We ignore the images where the *isCrowd* flag is true following the LayoutVAE [4] approach. The bounding boxes come from all 80 thing and 91 stuff categories. Our final dataset has 118280 layouts from COCO train split with a median length of 42

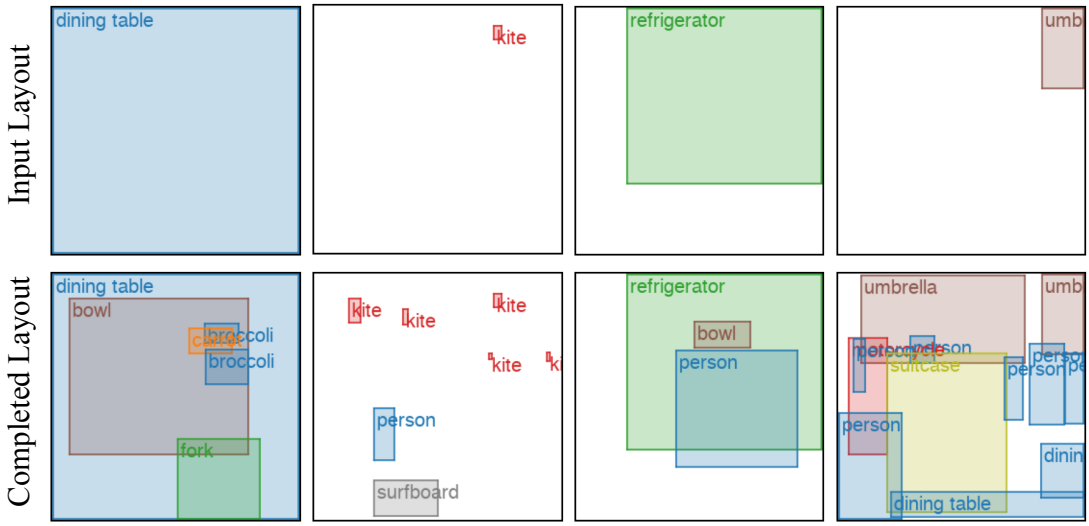


Figure 6.4: **Generated layouts.** Top row shows seed layouts input to the model. Bottom row shows the layout obtained with nucleus sampling. We skip the 'stuff' bounding boxes for clarity.

elements and 5000 layouts from COCO valid split with a median length of 33. We use the official validation split from COCO as test set in our experiments, and use 5% of the training data as validation.

Baseline Approaches. We compare our work with 4 previous methods - LayoutGAN[257], LayoutVAE [4], ObjGAN [267], and sg2im [6]. Since LayoutVAE and LayoutGAN are not open source, we implemented our own version of these baseline. Note that, like many GAN models, LayoutGAN was notoriously hard to train and our implementation (and hence results) might differ from author's implementation despite our best efforts. We were able to reproduce LayoutVAE's results on COCO dataset as proposed in the original paper and train our own models for different datasets. We also re-purpose ObjGAN and sg2im using guidelines mentioned in LayoutVAE.

Although evaluating generative models is challenging, we attempt to do a fair comparison to the best of our abilities. For our model, we keep architecture hyperparameters same across the

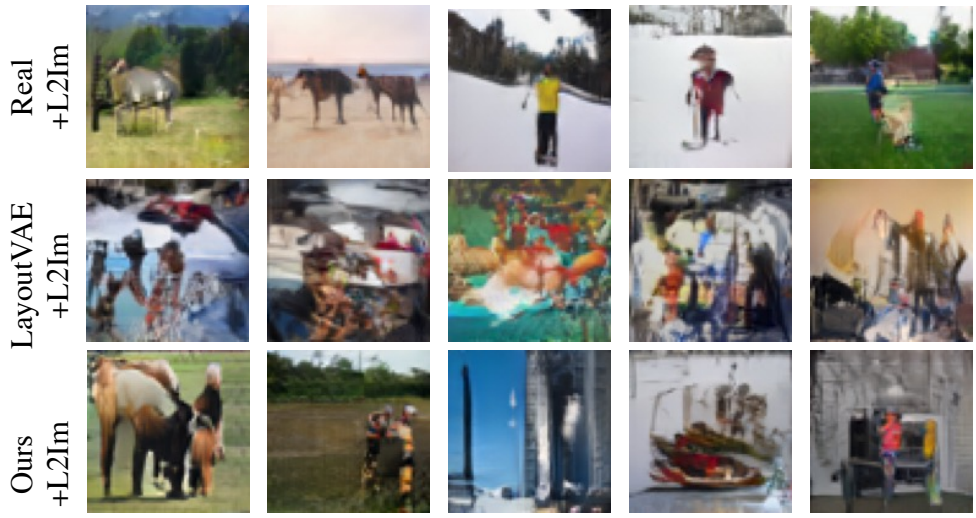


Figure 6.5: **Downstream task.** Image generation with layouts [5]. FID and IS scores for the generated images provided in Table 6.3.

datasets. For the baselines, we did a grid search over hyperparameters mentioned in the respective works & chose the best models according to validation loss. Some ablation studies are provided in the appendix.

Generated Samples. Fig. 6.4 shows layout completion task using our model on COCO dataset. Although the model is trained with all 171 categories, in the figure we only show 'thing' categories for clarity. We also use the generated layouts for a downstream application of scene generation [5].

Semantics Emerge via Layout. We posited earlier that capturing layout should capture contextual relationships between various elements. We provide further evidence of our argument in Fig. 6.6. We visualize the 2D-tsne plot of the learned embeddings for categories. We observe that super-categories from COCO are clustered together in the embedding space of the model. Certain categories such as window-blind and curtain (which belong to different super-categories) also appear close to each other. These observations are in line with observations made by [295] who use visual co-occurence to learn category embeddings. Table 6.2 shows word2vec [3] style analogies

being captured by embeddings learned by our model. Note that the model was trained to generate layouts and we did not specify any additional objective function for analogical reasoning task.

Finally, we also plot distribution of centers of bounding boxes for various categories in Fig. 6.7. $y-$coordinates of box centers are intuitive since categories such as 'sky' or 'airplane' are often on top of the image, while 'sea' and 'road' are at the bottom. This trend is observed in both real and generated layouts. $x-$ coordinates of bounding boxes are more spread out and do not show such a trend.

Quantitative evaluation. Following the approach of LayoutVAE, we compute negative log-likelihoods (NLL) of all the layouts in validation data using importance sampling. NLL approach is good for evaluating validation samples, but fails for generated samples. Ideally, we would like to evaluate the performance of a generative model on a downstream task. To this end, we employ Layout2Im [5] to generate an image from the layouts generated by each of the method. We compute Inception Score (IS) and Fréchet Inception Distance (FID) to compare quality and diversity of generated images. Our method is competitive with existing approaches in both these metrics, and outperforms existing approaches in terms of NLL.

Note that ObjGAN and LayoutVAE are conditioned on the label set. So we provide labels of objects present in the each validation layout as input. The task for the model is to then predict the number and postition of these objects. Hence, these methods have unfair advantage over our method and ObjGAN indeed performs better than our method and LayoutGAN, which are unconditional. We clearly outperform LayoutGAN on IS and FID metrics.

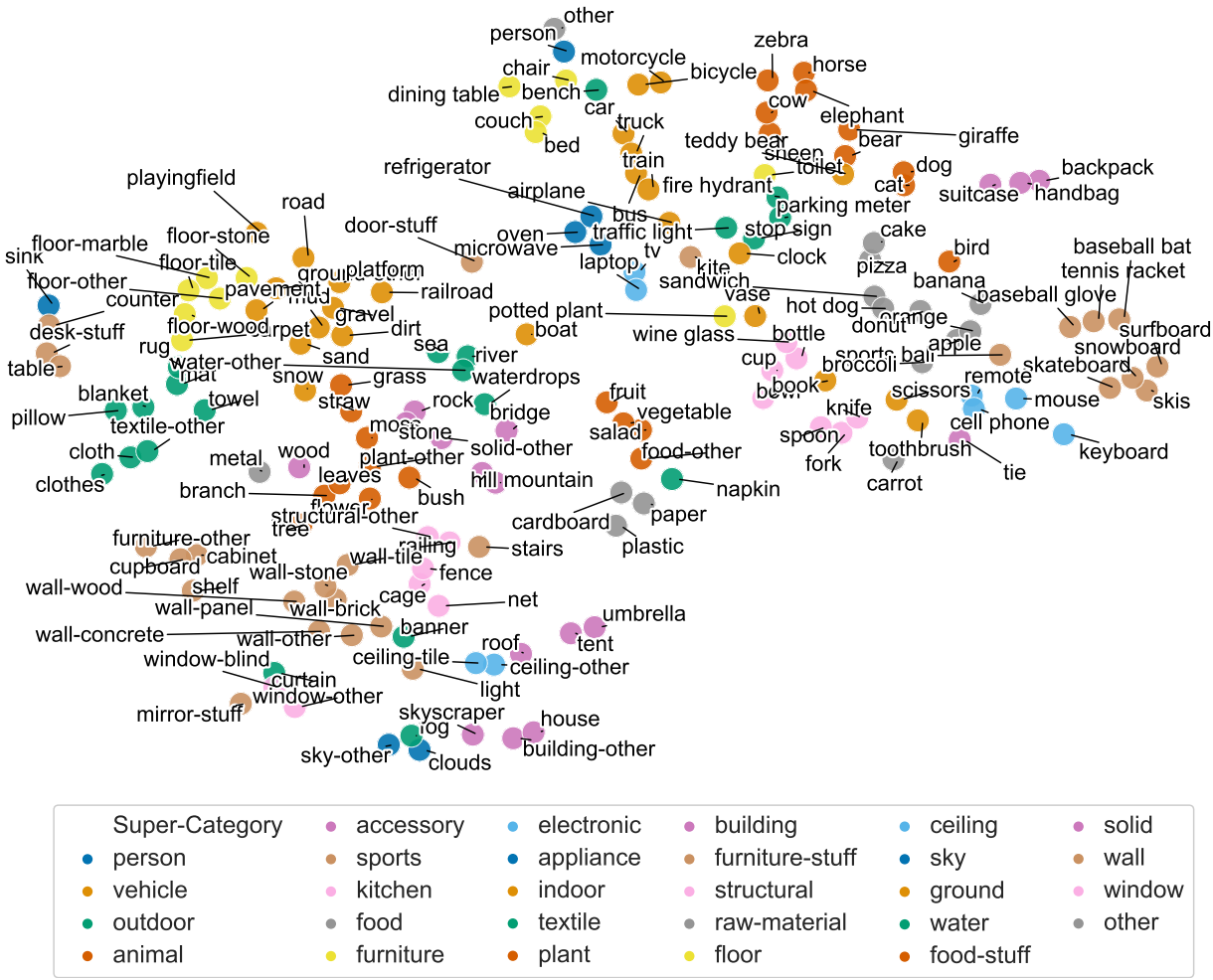


Figure 6.6: TSNE plot of learned category embeddings. Words are colored by their super-categories provided in the COCO. Observe that semantically similar categories cluster together. Cats and dogs are closer as compared to sheep, zebra, or cow.

6.3.3 Layouts for Apps and Documents

Rico Mobile App Wireframes. Rico mobile app dataset [258, 296] consists of layout information of more than 66000 unique UI screens from over 9300 android apps. Each layout consists of one or more of the 25 categories of graphical elements such as text, image, icon, *etc*.. A complete list elements is provided in the supplementary material. Overall, we get 62951 layouts in Rico with a median length of 36. Since the dataset does not have official splits, we use 5% of randomly

Table 6.2: **Analogies**. We demonstrate linguistic nuances being captured by our category embeddings by attempting word2vec [3] style analogies.

Analogy	Nearest neighbors
snowboard:snow::surfboard:?	waterdrops, sea, sand
car:road::train:?	railroad, platform, gravel
sky-other:clouds::playingfield:?	net, cage, wall-panel
mouse:keyboard::spoon:?	knife, fork, oven
fruit:table::flower:?	potted plant, mirror-stuff

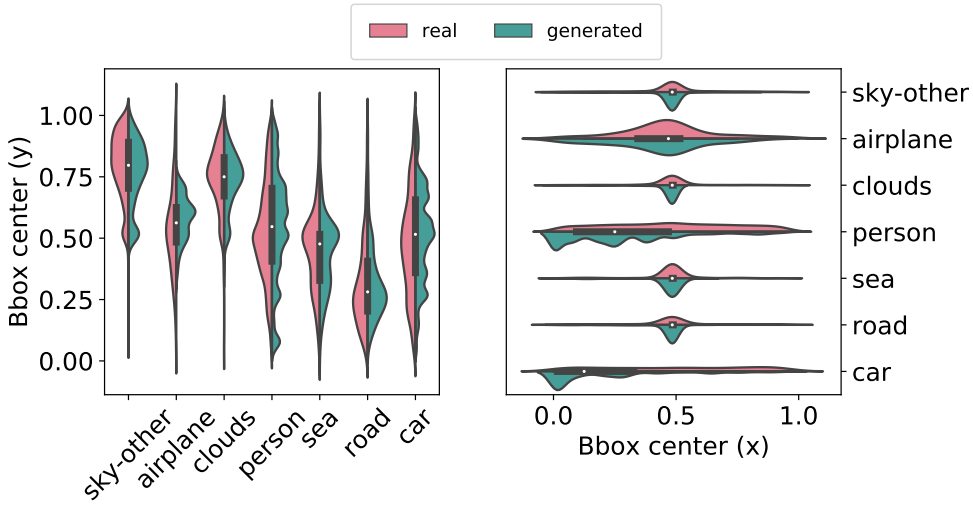


Figure 6.7: Distribution of xy-coordinates of bounding boxes centers. Distributions for generated and real layouts is similar. The y-coordinate tends to be more informative (*e.g.*, sky on the top, road and sea at the bottom)

selected layouts for validation and 15% for testing.

PubLayNet. PubLayNet [259] is a large scale document dataset consisting of over 1.1 million articles collected from PubMed Central. The layouts are annotated with 5 element categories - text, title, list, label, and figure. We filter out the document layouts with over 128 elements. Our final dataset has 335703 layouts from official train split with a median length of 33 elements and 11245 layouts from dev split with a median length of 36. We use the dev split as our test set and 5% of the training data for validation.

Generated layout samples. Fig. 6.8 and 6.10 shows some of the generated samples of our model

Table 6.3: **Quantitative Evaluations on COCO.** Negative log-likelihood (NLL) of all the layouts in the validation set (lower the better). We use the importance sampling approach described in [4] to compute. We also generated images from layout using [5] and compute IS and FID. Following [6], we randomly split test set samples into 5 groups and report standard deviation across the splits. The mean is reported using the combined test set.

Model	NLL ↓	IS ↑	FID ↓
LayoutGAN [257]	-	3.2 (0.22)	89.6 (1.6)
LayoutVAE [4]	3.29	7.1 (0.41)	64.1 (3.8)
ObjGAN [267]	5.24	7.5 (0.44)	62.3 (4.6)
sg2im [6]	3.4	3.3 (0.15)	85.8 (1.6)
Ours	**2.28**	**7.6 (0.30)**	**57.0 (3.5)**

from RICO mobile app wireframes and PubLayNet documents. Note that both the datasets share similarity in terms of distribution of elements, such as high coverage in terms of space, very little collision of elements, and most importantly alignment of the elements along both x and y-axes. Our method is able to preserve most of these properties as we discuss in the next section. Fig. 6.9 shows multiple completions done by our model for the same initial element.

Comparison with baselines. We use the same baselines discussed in §6.3.2. Fig. 6.10 shows that our method is able to preserve alignment between bounding boxes better than competing methods. Note that we haven't used any post-processing in order to generate these layouts. Our hypothesis is that (1) discretization of size/position, and (2) decoupling geometric fields in the attention module, are particularly useful in datasets with aligned boxes.

To measure this performance quantitatively, we introduce 2 important statistics. **Overlap** represents the intersection over union (IoU) of various layout elements. Generally in these datasets, elements do not overap with each other and Overlap is small. **Coverage** indicates the percentage of canvas covered by the layout elements. Table 6.4 shows that layouts generated by our method resemble real data statistics better than LayoutGAN and LayoutVAE.

Figure 6.8: **RICO layouts.** Generated layouts for the RICO dataset. We skip the categories of bounding boxes for the sake of clarity.

6.3.4 Failure Cases

Our model has a few failure cases, *e.g.*, in Fig. 6.3 in the third object (lamp), the parts are not connected - demonstrating a limitation of our approach arising from training the part auto-encoder and layout generator separately (and not jointly). Similarly, in 2D domains such as COCO, we observe that the model is biased towards generating high frequency categories in the beginning of the generation. This is illustrated in Fig. 6.7, which shows difference in distribution of real & generated layouts for persons and cars.

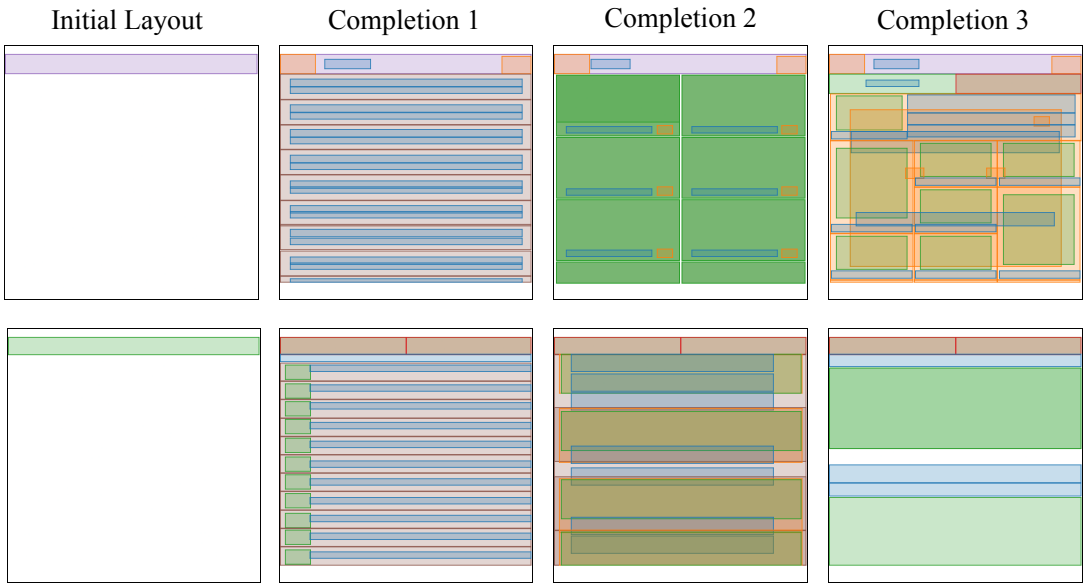


Figure 6.9: Multiple completions from same initial element

Table 6.4: Spatial distribution analysis for the samples generated using model trained on RICO and PubLayNet dataset. Closer the Overlap and Coverage values to real data, better is the performance. All values in the table are percentages (std in parenthesis)

	RICO			PubLayNet		
Methods	NLL↓	Coverage	Overlap	NLL↓	Coverage	Overlap.
sg2im [6]	7.43	25.2 (46)	16.5 (31)	7.12	30.2 (26)	3.4 (12)
ObjGAN [267]	4.21	39.2 (33)	36.4 (29)	4.20	38.9 (12)	8.2 (7)
LayoutVAE [4]	2.54	41.5 (29)	34.1 (27)	2.45	40.1 (11)	14.5 (11)
LayoutGAN [257]	-	37.3 (31)	31.4 (32)	-	45.3 (19)	8.3 (10)
Ours	**1.07**	33.6 (27)	23.7 (33)	**1.10**	47.0 (12)	0.13 (1.5)
Real Data	-	36.6 (27)	22.4 (32)	-	57.1 (10)	0.1 (0.6)

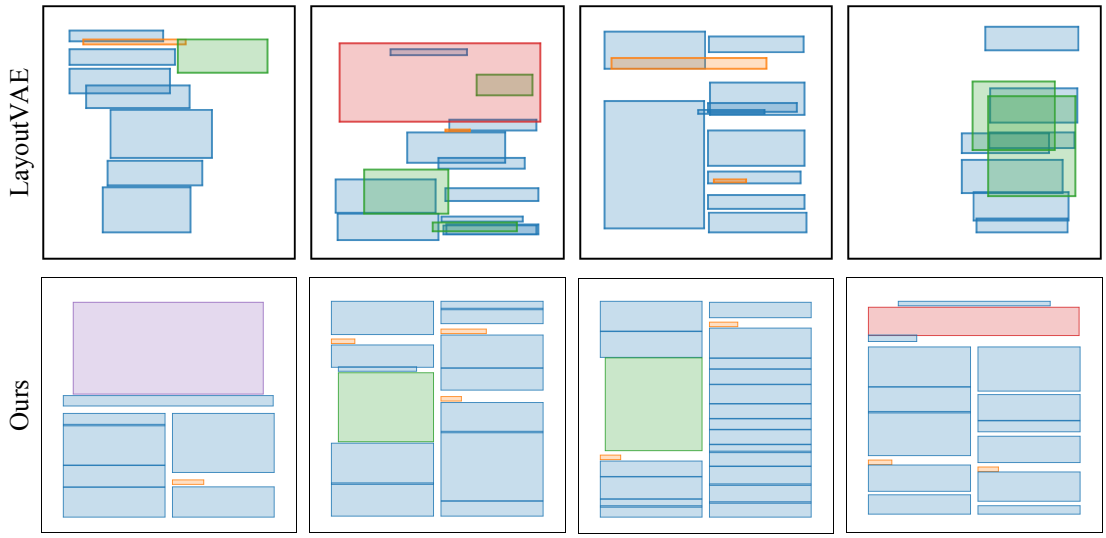


Figure 6.10: **Document Layouts.** Generated samples LayoutVAE (top) and our method (bottom). Our method produces aligned bounding boxes for various elements.

Chapter 7: Neural Space-filling Curves

In any form of digital communication, information is transmitted via a sequence of discrete symbols. This includes images and videos, even though they are inherently signals with two spatial dimensions (2D). The modus operandi for transmitting such signals is to (1) efficiently encode and quantize their values in the spatial or spectral domain, (2) linearize the signal to a one-dimensional (1D) sequence by using a standard scanning order such as raster, zig-zag, or Hilbert Curve order [297], and finally (3) apply a Shannon [298] style entropy coding technique such as Arithmetic coding [299] or Huffman coding [300] to further compress the 1D sequence. Given the ubiquity of images and videos in our lives, a large amount of effort has gone into optimizing each of these steps of digital communication. The focus of this work is the second step of linearizing the 2D spatial signal to a 1D sequence. A continuous scan order that traverses all spatial locations in two or higher dimensional signals exactly once is also known as the space-filling curve (SFC) [301].

Prior works have proposed various space-filling curves (SFCs) in the last hundred years, most of them context-agnostic, *i.e.*, they are completely defined by the size and dimension of the space without taking into account spatial information of the space, *e.g.*, pixels in the case of two-dimensional images. These universal context-agnostic SFCs are typically defined recursively to ensure simplicity and scale. Some of the SFCs also have spatial coherence properties and have

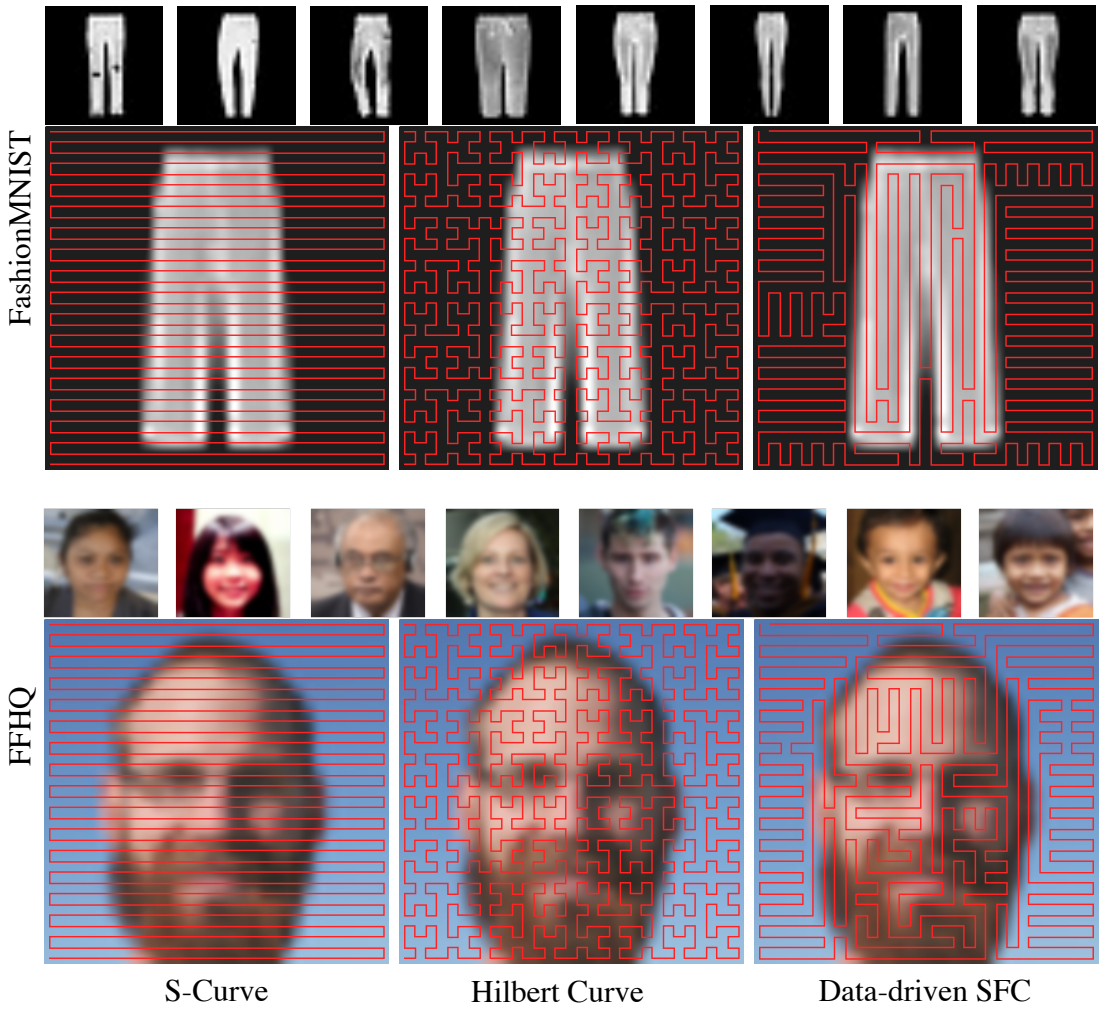


Figure 7.1: Given a set of images, a gif, or a video, Neural Space-filling Curves (SFC) can provide a more spatially coherent scan order for images as compared to universal scan orders such as S-curve, or Peano-Hilbert curves. As shown in the example of a trouser and a face, the scan line tends to cover the background before moving to the foreground. (SFCs generated here using half-resolution images and resized for clarity. Best viewed in color.)

been used in various image-based applications [302, 303, 304, 305].

However, in many applications such as video conferencing, health-care, or social media, the images being transmitted, are often repetitive with a similar layout and content with minor variations transmitted over and over again. GIFs are another great example that consists of highly repetitive content and need to be stored efficiently and often, losslessly. Since universal SFCs do not utilize the intrinsic information of image content, they are far from optimal for a single image

or a set of images with repetitive structure (refer to Fig. 7.1 for an example). Dafner *et al.*[306] proposed SFCs that exploit the inherent correlation between pixel values in an image. Our work improves upon Dafner *et al.*, in two aspects.

- Instead of discovering a single SFC for every image independently, we propose a data-driven technique to find optimal SFCs for a set of images. We postulate that context-based SFCs are more suitable for linearizing a group of images (or a short video/gif), since the cost of storing the SFC itself can be amortized by the number of images.
- We devise a novel alternating minimization technique to train an SFC weights generator, which allows us to optimize for any given objective function, even when not differentiable.

To the best of our knowledge, ours is the first work to propose a machine learning method for computing context-based SFCs and opens new directions for future research on optimal scanning of 2D and 3D grid-based data structures such as images, videos, and voxels. We demonstrate both quantitatively and qualitatively the benefit of our approach in various applications.

7.1 Related Work

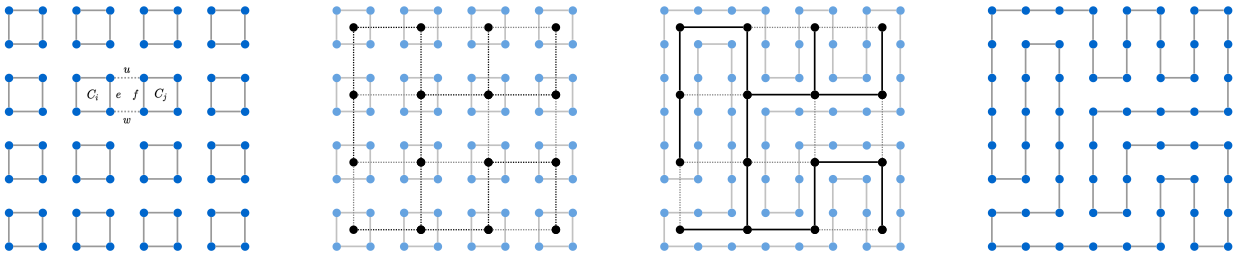
Space-filling Curves (SFCs), introduced by Peano in 1890 [301] and Hilbert in 1891 [297], are injective functions that map a line segment to a continuous curve in the unit square, cube, or hypercube. Most classic SFCs such as Peano-Hilbert, Sierpinski [307], and Moore [308] curves are defined recursively which allows them to scale up to arbitrary resolution with a number of favorable properties. In fact, [309] showed that the entropy of this pixel sequence asymptotically converges to the two-dimensional entropy of the original image for a large number of

images coming from sufficiently random sources. Because of their self-organizing capabilities, Hilbert curves have found applications in compression [302, 303, 304, 305], computing [310], recognition [311, 312], security [313], databases [314], electronics [315], biology [316] and even web-comics [317]. Hilbert curves have also been used to solve multidimensional task allocation problems in parallel processing [318]. This scheme is used in many job schedulers, such as the famous SLURM [319].

Dafner *et al.*[306] proposed the first context-based SFCs. Their work makes use of cover and merge algorithm [313] to compute SFC for an image. Ouni *et al.*[320] employ image gradient based method to compute context-based SFCs, and they apply their SFCs to lossless compression tasks. Zhou *et al.*[321] improve Dafner *et al.*'s method by considering both data values and location coherency. They also generalize their method to multiscale data via quadtrees and octrees.

However, computing a unique SFC specific to an image has limited applications. Compression techniques such as LZW [309] that exploit the correlation between nearby pixel values during encoding, for example, can do a better job with context-based SFCs, however, the cost of storing an SFC itself for each image gives away the advantage. Universal SFCs, such as Hilbert curves don't have this disadvantage and are frequently used in compression applications. In this work, we argue that, applications that need to store and transmit images with repetitive structures such as gifs, can benefit greatly from context-based SFC optimized for that set of images.

Finding an optimal SFC is a combinatorial optimization problem. Solving combinatorial problems have a rich history in the field of machine learning. In 1985, [322] first attempted to solve these problems using a neural network. In the deep learning era, various flavors of attention have been proposed [263, 323, 324] to reach an approximate solution of NP-hard problems in



(a) Cover $\mathcal{G}$ (b) Dual $\mathcal{G}'$ (c) $\mathcal{T}$: MST of $\mathcal{G}'$ (d) Merged

Figure 7.2: **Cover and Merge Algorithm.** (a): An 8×8 image fully covered by 2×2 circuits. (b): The dual graph $\mathcal{G}'$ (black) built on the covering circuits $\mathcal{G}$ (blue). (c): The Minimum Spanning Tree $\mathcal{T}$ (black solid lines) of $\mathcal{G}'$ (all black lines) and the Hamiltonian Circuit (blue) induced by $\mathcal{T}$. (d): A single Hamiltonian circuit merged from the covering circuits. See more details in Sec. 7.2.1.

computer science. Since combinatorial optimization problems are inherently non-differentiable, reinforcement learning (RL) is also a promising alternative for addressing these problems [263, 324, 325, 326]. In our initial experiments, we found the performance of RL approach in our use-case to be unstable and inconsistent.

7.2 Approach

We first describe the algorithm for computing SFC for one image as proposed by Dafner *et al.*[306] in Section 7.2.1. We then extend this treatment to a more general setting where we can optimize the SFC for any non-differential objective function for multiple images in Section 7.2.2. The rest of the Section 7.2 describes major components of our framework and training procedure in detail.

7.2.1 Overview of Dafner *et al.*(Single Image-based SFC)

Given an image, [306] represents it as an undirected graph $\mathcal{G}$ whose nodes are pixel locations, and each pixel is connected to its 8 neighboring pixels by an edge. Generating a context-based SFC from the given image is then equivalent to finding a Hamiltonian path in graph $\mathcal{G}$.

They use the cover and merge algorithm (initially proposed by Matias *et al.*[313] to scramble a video signal for secure transmission). As the name suggests, the algorithm works by finding a Hamiltonian path for the image-grid graph $\mathcal{G}$ in two steps - **cover** and **merge**.

In the **cover** step, a dual undirected graph $\mathcal{G}'$ is constructed from $\mathcal{G}$. The vertices of $\mathcal{G}'$ are small disjoint square circuits covering the whole of $\mathcal{G}$ as shown in Figure 7.2a. Each circuit covers 4 pixels. We call these initial circuits $C_1, C_2, ..., C_k$ and connect (C_i, C_j) if the circuits C_i and C_j are adjacent in the original graph $\mathcal{G}$. Figure 7.2b shows a dual graph example built for an 8×8 image.

In the **merge** step, all circuits are merged to form a single Hamiltonian circuit. To merge the circuits optimally, a weight is assigned to each edge (C_i, C_j) in $\mathcal{G}'$ representing the "cost" of merging circuits C_i and C_j. The weight $w(C_i, C_j)$ of the edge connecting circuits C_i and C_j is defined as the cost of exchanging the edges e and f with the edges u and w in the image graph:

$$w(C_i, C_j) = |u| + |w| - |e| - |f|, \tag{7.1}$$

where $|\cdot|$ corresponds to the absolute difference in pixel values at the two vertices of the edge in $\mathcal{G}$. A minimum spanning tree $\mathcal{T}$ is then constructed using these weights. Figure 7.2c shows what $\mathcal{T}$ might look like for $\mathcal{G}'$. Next, we start merging circuits that are part of $\mathcal{T}$. Merging two circuits corresponds to removing their adjacent edges in $\mathcal{G}$ (*e.g.*, edge e and f in Figure 7.2a), and creating new edges (*e.g.*, edge u and w in Figure 7.2a). Note that only adjacent circuit pairs can be merged. Given the spanning tree, all merging operations on $\mathcal{G}$ can be done in a linear time to obtain a Hamiltonian circuit as in Figure 7.2d. Finally, the SFC or Hamiltonian path can be obtained by cutting the circuit at an arbitrary point.

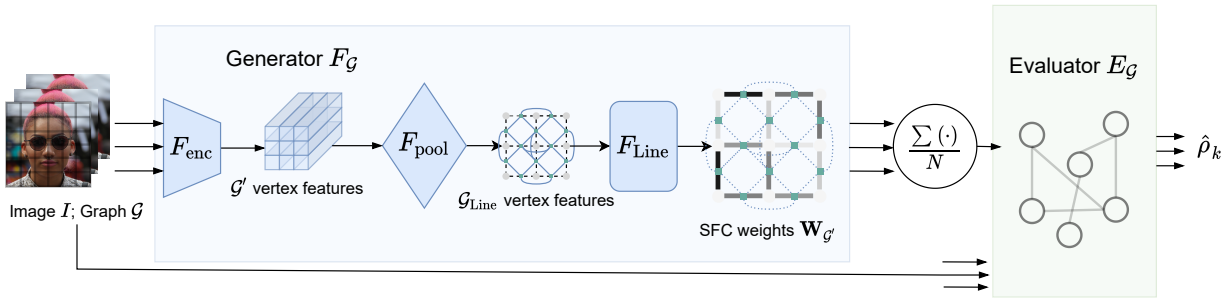


Figure 7.3: **Neural SFC pipeline.** The Neural SFC model consists of a weight generator $\mathrm{F}_{\mathcal{G}}$ and a weight evaluator $\mathrm{E}_{\mathcal{G}}$. Taking one or more images as input, $\mathrm{F}_{\mathcal{G}}$ extracts the deep features of the image and generates the SFC weights. $\mathrm{E}_{\mathcal{G}}$ takes both the image(s) and the corresponding SFC weights as input $\mathrm{E}_{\mathcal{G}}$ then estimates the negative autocorrelation of the image pixel sequence inferred by the input weights to evaluate the goodness of the input weights with respect to the input image. See more details in Sec. 7.2.2.

7.2.2 Our Approach

Dafner's approach [306] is effective at exploiting the local relationships in an image. However, it has a few limitations.

- Since the edge weights are computed using only two adjacent pixels, the receptive field is limited, and the resulting SFC does not take into account the long-range context in the image.
- Dafner's approach only works for one image at a time. The notion of context-based SFCs can be further generalized to finding an SFC for a set of images.
- In the current form, [306] cannot optimize for arbitrary objective functions. The context-based SFC obtained is closely tied to edge weights defined by Equation (7.1) which encourage autocorrelation with lag-2 in the pixel sequence obtained.

We address each of these issues in our data-driven approach to infer an optimal SFC for a set of images for any objective function. For brevity, we will optimize for lag-k autocorrelation, however, in our experiments, we will show how we can modify this objective and directly

optimize for a downstream application such as compression.

Setup. For the remainder of this section, we use the following notation. We are given a set of N images. Each image (color or grayscale) has a resolution of $H \times W$. For a given image I, we define graph $\mathcal{G}$ over its HW pixel locations. The dual graph $\mathcal{G}'$ consists of 2×2 disjoint circuits covering all the vertices of $\mathcal{G}$ as defined in the previous section. We first note that the whole cover and merge algorithm is context-agnostic barring the step where we assign weights to the edges of the dual graph. Therefore, a context-based SFC can be completely parameterized by these weights, denoted by $\mathbf{W}_{\mathcal{G}'}$. For a given set of weights of image I, the merge operation provides us with a Hamiltonian circuit. A Hamiltonian path, or an SFC, can be obtained by breaking the merged Hamiltonian circuit at any point. Hence, the problem of finding an SFC can be reduced to finding the optimal weights $\mathbf{W}_{\mathcal{G}'}$. We propose to learn a neural network, $\mathrm{F}_{\mathcal{G}}$, to approximate $\mathbf{W}_{\mathcal{G}'}$. Once we have the optimized weights, the merge operation is fast and efficient in terms of both memory and speed, and we exploit it to get our desired SFC.

7.2.3 Weight Generator

The weight generator $\mathrm{F}_{\mathcal{G}}$ is designed to take as input a single image I, and output the edge weights $\mathbf{W}_{\mathcal{G}'}$ of its dual graph, *i.e.*, $\mathbf{W}_{\mathcal{G}'} = \mathrm{F}_{\mathcal{G}}(I)$. While the input dimension is $H \times W$, the output $\mathbf{W}_{\mathcal{G}'}$, has a size equal to the number of edges in the dual graph $\mathcal{G}'$. It is trivial to show that for the dual graph $\mathcal{G}'$ consisting of $\frac{H}{2} \times \frac{W}{2}$ vertices (or circuits), the corresponding number of edges will be $\frac{HW-H-W}{2}$. Further, the edges of $\mathcal{G}'$ do not conform to a 2D grid structure as the

input image I. To address this, we decompose the weight generator $\mathrm{F}_{\mathcal{G}}$ further in three modules.

$$\mathbf{W}_{\mathcal{G}'} = \mathrm{F}_{\mathcal{G}}(I) = \mathrm{F}_{\text{Line}} \circ \mathrm{F}_{\text{pool}} \circ \mathrm{F}_{\text{enc}}(I). \tag{7.2}$$

Figure 7.3 shows the architecture of the weight generator. The first submodule is a dual graph encoder F_{enc}, which takes I as the input and extracts a deep representation of the vertices of dual graph $\mathcal{G}'$, resulting in a $\frac{H}{2} \times \frac{W}{2} \times d$ dimensional output, where d is the number of output feature maps. In this work, F_{enc} is implemented using a fully convolutional neural network with residual connections.

The second submodule F_{pool} consists of two pooling filters (and no trainable weights) of dimension 1×2 and 2×1 applied sequentially. F_{pool} imitates the graph operations by aggregating the features of vertices, and hence forming edge features of $\mathcal{G}'$. Given a d-dimensional representation of edges, we want to compute a scalar weight for each edge in $\mathcal{G}'$. It is desirable that this scalar weight can exploit not only the edge features, but also long-range relationships among the edges. Hence we construct a Line graph [327] $\mathcal{G}_{\text{Line}}$ to represent the adjacency between edges of $\mathcal{G}'$. Each vertex in $\mathcal{G}_{\text{Line}}$ corresponds to an edge in $\mathcal{G}'$. For every two edges in $\mathcal{G}'$ that have a vertex in common, there is an edge between their corresponding vertices in $\mathcal{G}_{\text{Line}}$.

Using the Line graph, the edge features of $\mathcal{G}'$ becomes the vertex feature of $\mathcal{G}_{\text{Line}}$, and the adjacent relations of edges of $\mathcal{G}'$ are represented by edges of $\mathcal{G}_{\text{Line}}$. To compute the scalar weights on $\mathcal{G}_{\text{Line}}$, we introduce the third submodule, a weights regressor F_{Line} to run on $\mathcal{G}_{\text{Line}}$. F_{Line} can be implemented using Graph Neural Network modules. In this work, we use GCN [328] in MNIST experiments and GAT [329] in FFHQ Faces experiments.

7.2.4 Objective Functions

The weight generator described above can generate edge weights for a given image. For every mini-batch of images, we take an expected value of the weights for all the images to get $\overline{\mathrm{W}}_{\mathcal{G}'}$. Given $\overline{\mathrm{W}}_{\mathcal{G}'}$, we can compute an SFC for the mini-batch of images. The quality or 'goodness' of an SFC can be different for different applications. In this paper, we consider two plausible objectives: the 1D autocorrelation and the LZW sequence length. For both of them, the first step is to flatten the given image I to the 1D pixel sequence $\{y_i\}$ based on the SFC defined by $\overline{\mathrm{W}}_{\mathcal{G}'}$.

Autocorrelation. The 1D autocorrelation measures the internal local similarity of a 1D sequence. Therefore, the smaller the 1D autocorrelation is, the better the SFC is. The lag-k 1D autocorrelation of a pixel sequence $\{y_i\}$ of length HW is defined as

$$\rho_k = \frac{\sum_{i=1}^{HW-k} y_i y_{i+k}}{\sum_{i=1}^{HW} y_i^2}. \tag{7.3}$$

Code Length. Another SFC quality metric, the LZW sequence length, is inspired from the Lempel-Ziv Welch (LZW) encoding [330, 331], which is popularly used to encode GIFs losslessly. Its performance depends on the amount of redundant data in a given sequence. Given a pixel sequence $\{y_i\}$, it's LZW length is defined as

$$L = \mathrm{length}(\mathrm{Encode}(\{y_i\})), \tag{7.4}$$

where Encode is the LZW-encoding function, and length measures the length of a sequence.

Note that computing ρ_k or L requires us to first obtain a minimum spanning tree to infer

the SFC from I and $\overline{\mathrm{W}}_{\mathcal{G}}$, which is a non-differentiable operation. Therefore, these metrics, by themselves, cannot be directly used to optimize a neural network. Hence we can't simply backpropagate gradient information to update $\mathrm{F}_{\mathcal{G}}$.

To overcome the problem of non-differentiability of SFC computation, we train an evaluator neural network, $\mathrm{E}_{\mathcal{G}}$, as a differentiable proxy to estimate the resulting autocorrelation of SFC weights computed by $\mathrm{F}_{\mathcal{G}}$ from the context image(s). By carefully designing the training procedure, our model $\mathrm{E}_{\mathcal{G}}$ is able to approximate any non-differentiable metric, hence serving as an effective loss function of the weight generator. Figure 7.3 summarizes our approach.

Also note that, while we refer to the lag-k autocorrelation and the LZW length as our objective functions, we can replace them with any loss (or reward) suitable for a given application, even if it is not differentiable.

7.2.5 Weight Evaluator

The weight evaluator, denoted by $\mathrm{E}_{\mathcal{G}}$, acts as a differentiable proxy to estimate the objective Φ. Depending on the task, we choose $\Phi = -\rho_k$ for minimizing the negative autocorrelation or $\Phi = L$ for minimizing LZW length.

Given the average SFC weights $\overline{\mathrm{W}}_{\mathcal{G}}$ and an image I, we compute

$$\hat{\Phi} = \mathrm{E}_{\mathcal{G}}(\overline{\mathrm{W}}_{\mathcal{G}}, I), \tag{7.5}$$

$$\mathcal{L}_{\mathrm{E}} = \mathbb{E}\left[\|\Phi - \hat{\Phi}\|\right], \tag{7.6}$$

where $\mathcal{L}_{\mathrm{E}}$ denotes the expected value of l-2 error between groundtruth lag-k autocorrelation and the predicted autocorrelation by $\mathrm{E}_{\mathcal{G}}$ computed for the entire batch. $\mathcal{L}_{\mathrm{E}}$ serves as the objective

function for training the evaluator.

The inputs to the weight evaluator $\mathrm{E}_{\mathcal{G}}$ are $\overline{\mathrm{W}}_{\mathcal{G}'}$ and I. Similar to Sec. 7.2.3, we again decompose the weight evaluator $\mathrm{E}_{\mathcal{G}}$ into submodules,

$$\hat{\Phi} = \mathrm{E}_{\mathcal{G}}(\overline{\mathrm{W}}_{\mathcal{G}'}, I) = \mathrm{E}_{\text{Line}}(\overline{\mathrm{W}}_{\mathcal{G}'} \,\|\, \mathrm{E}_{\text{pool}} \circ \mathrm{E}_{\text{enc}}(I)), \tag{7.7}$$

where $\|$ denotes the concatenation along the feature dimension. Following the graph encoding procedure of the weight generator $\mathrm{F}_{\mathcal{G}}$, E_{pool} and E_{enc} are functionally identical to F_{pool} and F_{enc}, respectively. The final submodule E_{Line}, which takes similar input to F_{Line}, has a different head to predict the estimated objective value $\hat{\Phi}$. The backbone of E_{Line} is also implemented using a GCN or GAT, followed by an average pooling operation and a simple MLP to predict $\hat{\Phi}$ as a single value.

7.2.6 Training

We adopt an alternating optimization procedure for training the core components of our architecture $\mathrm{F}_{\mathcal{G}}$ and $\mathrm{E}_{\mathcal{G}}$. Algorithm Algorithm 1 gives an overview of the training schema. The weight evaluator $\mathrm{E}_{\mathcal{G}}$ solves the regression task of computing the estimated objective $\hat{\Phi}$ for a given set of SFC weights $\overline{\mathrm{W}}_{\mathcal{G}'}$. Given the context image I and SFC weights $\overline{\mathrm{W}}_{\mathcal{G}'}$, we can get the groundtruth objective by first running Prim's algorithm [332] to get the desired SFC followed by Equation (7.3) or Equation (7.4).

Once we have the groundtruth Φ, we optimize a standard L2 loss commonly used in regression methods to train $\mathrm{E}_{\mathcal{G}}$, as we described in Equation (7.6). Since we eventually need to use $\mathrm{E}_{\mathcal{G}}$ to train the upstream network F, we would like $\mathrm{E}_{\mathcal{G}}$ to be trained on a diverse range of input SFC

Algorithm 1: Training Neural SFC

Data: A set of N images each of resolution $H \times W$
Result: SFC weights $\overline{\mathrm{W}}_{\mathcal{G}'}$ for the image set

Randomly initialize $\mathrm{F}_{\mathcal{G}}$ and $\mathrm{E}_{\mathcal{G}}$;
repeat
  `// training` $\mathrm{F}_{\mathcal{G}}$
  Sample a minibatch of B images;
  Forward pass for the weight generator $\mathrm{F}_{\mathcal{G}}$, $\mathbf{W}_{\mathcal{G}'} \leftarrow \mathrm{F}_{\mathcal{G}}(I)$;
  Expected weights for the mini-batch $\overline{\mathrm{W}}_{\mathcal{G}'} \leftarrow \mathbb{E}(\mathbf{W}_{\mathcal{G}'})$;
  Forward pass for the weight evaluator $\hat{\Phi} \leftarrow \mathrm{E}_{\mathcal{G}}(\overline{\mathrm{W}}_{\mathcal{G}'}, I)$;
  SGD update for $\mathrm{F}_{\mathcal{G}}$ keeping $\mathrm{E}_{\mathcal{G}}$ fixed, $\nabla_{\mathrm{F}_{\mathcal{G}}} \mathbb{E}\left[\hat{\Phi}\right]$;

  `// training` $\mathrm{E}_{\mathcal{G}}$
  For each example in B, with a probability p_1, get $\mathbf{W}_{\mathcal{G}'}$ using Equation (7.1), with a probability p_2, sample $\mathbf{W}_{\mathcal{G}'} \sim \mathcal{N}(0,\, 1)$ and with a probability $1 - p_1 - p_2$, keep $\mathbf{W}_{\mathcal{G}'} = \mathrm{F}_{\mathcal{G}}(I)$;
  Run a forward pass for the weight evaluator $\hat{\Phi} \leftarrow \mathrm{E}_{\mathcal{G}}(\overline{\mathrm{W}}_{\mathcal{G}'}, I)$;
  For the whole batch, compute the ground truth Φ using Equation (7.3) or Equation (7.4);
  SGD update for $\mathrm{E}_{\mathcal{G}}$ with $\nabla_{\mathrm{E}_{\mathcal{G}}} \mathcal{L}_{\mathrm{E}}$;
until $\mathcal{L}_E \to 0$;

weights $\mathbf{W}_{\mathcal{G}'}$.

The $\mathrm{F}_{\mathcal{G}}$ can be trained trivially using a fixed $\mathrm{E}_{\mathcal{G}}$. We empirically observed that training them alternately improves the training dynamics thus boosting the SFC quality.

7.3 Experiments

In this section, we evaluate the ability of the proposed training scheme to generate optimized Space-filling Curves (SFC) for a set of images. We further validate the efficacy of the Neural SFCs on real-world applications such as image or gif compression. We compare with standard Raster scan and Hilbert curves. Note that even though Raster scan is not an SFC mathematically speaking, we use it for benchmarking in our experiments due to its prevalence.

7.3.1 Datasets

We trained the Neural SFC model on four different datasets. Both **MNIST** [333] and **Fashion-MNIST** [334] comprise of 60000 training images, and 10000 test images. Each image is a 28×28 grayscale image, which we resize to 32×32 to do a fair comparison with Hilbert curves which can be defined only when the image resolution is a power of 2. Resizing is done by a simple zero padding around the image. We observe a lot of intra-class similarity in the case of both MNIST and Fashion-MNIST, *i.e.*, the images within the same class are similar in layout and content to each other, and hence we train a single SFC for each MNIST and Fashion-MNIST class.

We also consider **FFHQ** [248] dataset. We downsample all the images to size 32×32 using bilinear interpolation. FFHQ is a dataset of celebrity faces and contains less noise compared to datasets like CelebA [335]. We split the dataset into 60000 training and 10000 test images. We train a single SFC for all the images in the data.

Lastly, in order to demonstrate a real-world application of SFCs designed for a set of images, we use a large-scale GIF dataset **TGIF** [336]. The dataset consists of 80,000 training gifs, and 11,360 test gifs. We train a single Neural SFC model that takes a gif as an input and outputs an optimized SFC for the gif. We evaluate it on every gif in the test dataset. Average numbers are reported.

7.3.2 Training details

We consider two different objective functions for training Neural SFC. First, in order to compare our method with Dafner *et al.*[306], we train our model using the autocorrelation objec-

tive function. Since Dafner's method cannot generate SFCs for an image-set trivially, we compute Dafner's image-set SFC by taking the expected value of $\mathbf{W}_{\mathcal{G}'}$ defined in their method for all the images in the training set. Another possible choice is to calculate an average image for the entire image set, then run Dafner's method on it to generate an image-set SFC. We use the first setting in all experiments in our main paper because it empirically performs better. In all our experiments, we set $k = 6$, such that lag-6 autocorrelation is used to train the weight evaluator $\mathrm{E}_{\mathcal{G}}$.

Second, we also provide quantitative results on the compression of images and gifs using a Lempel-Ziv encoder (LZW) encoding scheme. Specifically, we optimize the SFC separately for each gif, although, it is possible to train a large SFC encoder on the entire gif dataset. We leave the study for evaluating context-based SFC for large- gifs or video datasets for future work.

7.3.3 Qualitative Evaluation

Visualizing the 1D sequence. In Figure 7.4, we show a few examples to illustrate the difference between Neural SFCs and Hilbert Curve on the MNIST, Fashion-MNIST, and FFHQ images. We look at the SFC (red curve overlayed on the image of the digits/clothing/faces). While the Hilbert Curve outputs the same SFC for any 32×32 image in the world, Neural SFC optimizes the SFC for a given set of images. To see this difference, we flatten the SFCs into a 1D sequence (images on the right of the digits/clothing/faces), and observe that Neural SFCs tend to keep better long-range spatial coherence than Hilbert Curve. In both cases, Neural SFCs show pixels in fewer clusters as compared to Hilbert curves. Specifically, Neural SFCs are able to roughly stay in the bright regions until they all get covered. Therefore, bright pixels will mostly gather in one contiguous segment in the 1D sequence inferred from a Neural SFC. In contrast, Hilbert

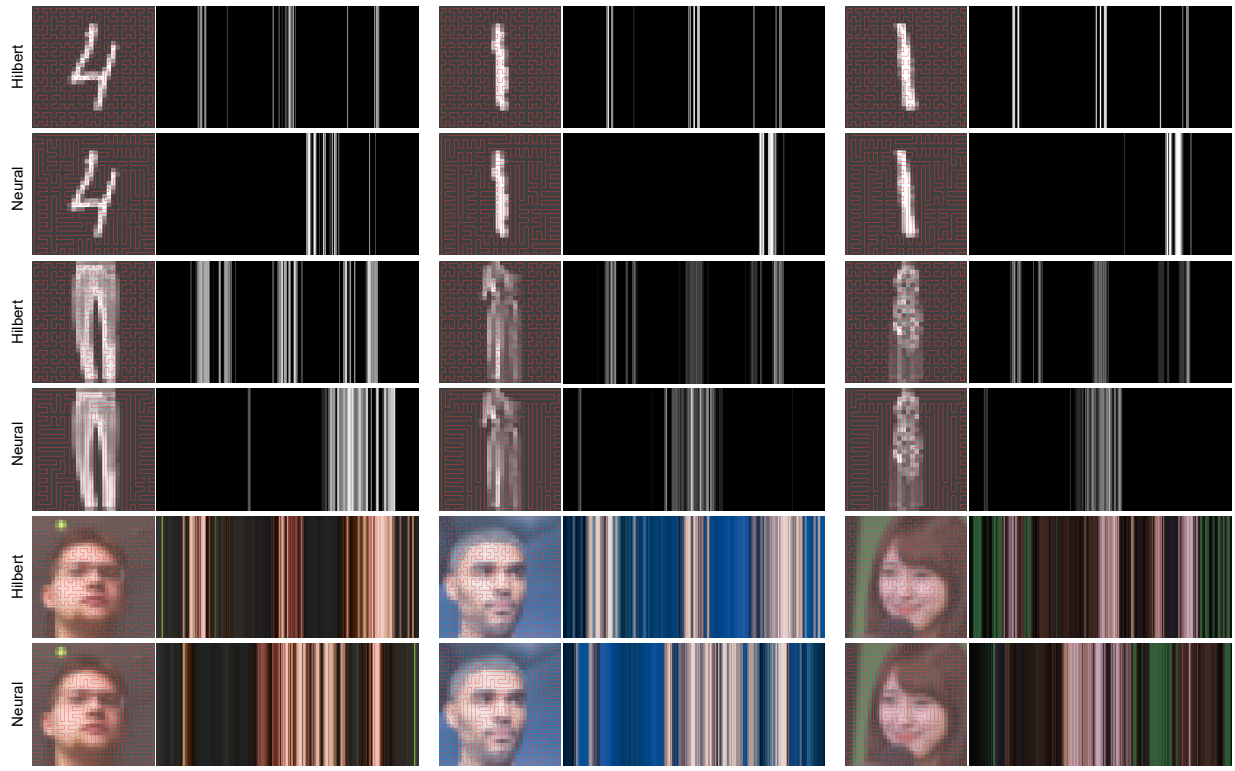


Figure 7.4: Qualitative comparison between Hilbert curves and Neural SFCs. Left: SFC (in red color) overlayed on the image. Right: Image flattened according to the SFC and visualized in 1-dimension. Images in the top two rows are from MNIST, the ones in the middle two rows are from Fashion-MNIST, and the ones in the bottom two rows are from FFHQ Faces. Neural SFCs on images from MNIST and Fashion-MNIST are class-conditional, *i.e.*, computed for each class. Therefore, for MNIST and Fashion-MNIST, Neural SFCs in the right two columns are the same since the two images have the same class label. In all datasets, Neural SFCs are more spatially coherent and produce fewer clusters when visualized in 1-dimension. Best viewed in color.

curves often result in multiple clusters of contiguous structures in the 1D sequence. We provide more examples in the supplementary material.

SFCs obtained with different objectives. Figure 7.5 shows two different SFCs obtained by our approach when trained with different objective functions. The figure on the left corresponds to SFC optimized for LZW encoding length, while the figure on the right corresponds to SFC optimized for auto-correlation. We observe that generally, SFCs optimized for LZW encoding length are better at short-lag auto-correlation (as compared to SFCs optimized for lag-6 as in most of our experiments). This results in an SFC with fewer turns and straighter paths.

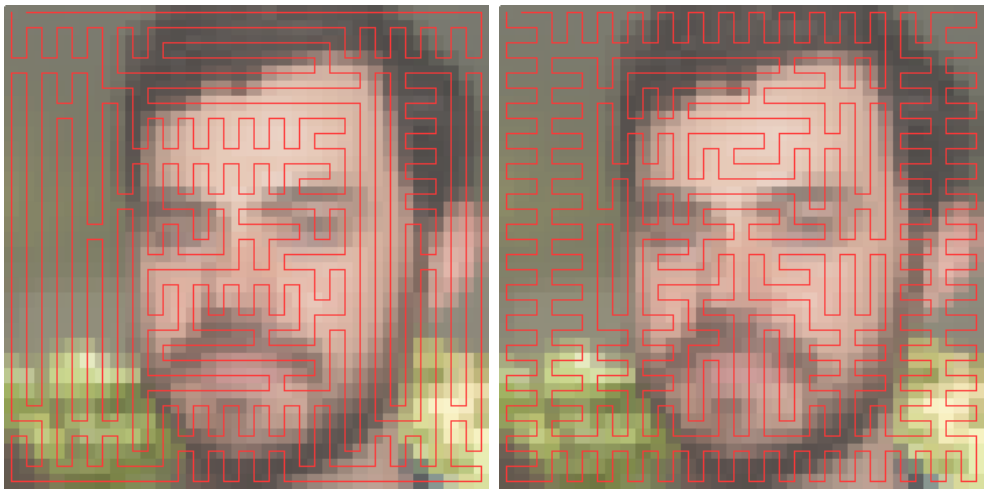
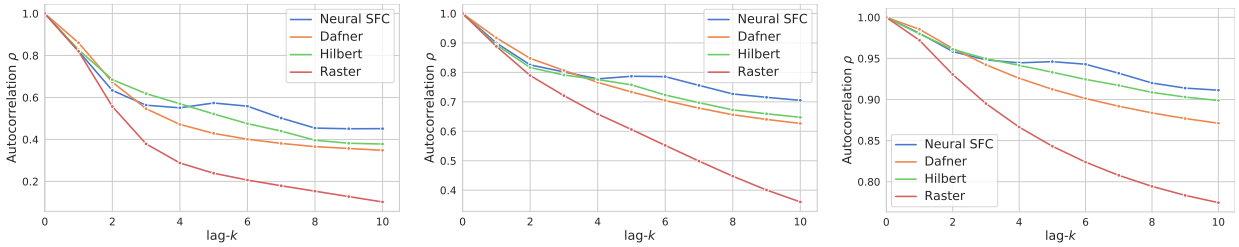


Figure 7.5: We visualize the Neural SFCs training with two objectives considered in this paper – autocorrelation and LZW encoding.



(a) MNIST (b) Fashion-MNIST (c) FFHQ 32×32

Figure 7.6: lag-k autocorrelation for MNIST, Fashion-MNIST and FFHQ datasets. While Dafner *et al.*provide higher autocorrelation for small lag, *i.e.*, from $k = 2$ to $k = 4$, Neural SFCs outperform Dafner *et al.*for $k > 4$ in all the datasets. Note that we trained our model for $k = 6$, and hence this behaviour is expected.

7.3.4 Optimizing Autocorrelation

To quantitatively evaluate the generated SFCs, we plot lag-k autocorrelations of pixel sequences obtained from test sets of three different datasets MNIST, Fashion-MNIST, and FFHQ. Note that even though, we trained our models only to optimize the lag-6 autocorrelation, we plot them for a range of values of lag-k. From Figure 7.6, we first observe that the trend is somewhat consistent across multiple datasets, even though they are very different in nature. Neural SFC

performs the same or slightly worse than other SFCs at lower values of k. However, for $k > 4$, it outperforms other SFCs by a wider margin. This is intuitive since our model was optimized to increase the autocorrelation for a large value of lag. This also reflects our model's ability to capture long-range and global information. We believe that higher gains can be obtained if we train and test using the same k value.

In the second and the third columns of Table 7.1, we show the autocorrelation values for MNIST, Fashion-MNIST, and FFHQ. We observe that the performance of Dafner's SFCs is even worse than the Hilbert Curve in most cases. This indicates that the naive average of several good SFCs may not be a good way to compute the SFC for a set of images.

7.3.5 Optimizing Code Length

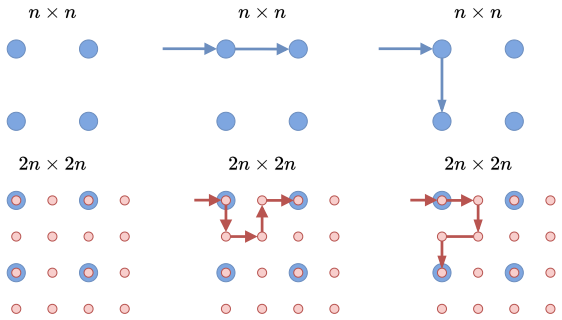
In this section, we study how Neural SFC can improve image compression results. Specifically, we use the LZW length objective to optimize $\mathrm{E}_{\mathcal{G}}$. This means we set $\Phi = L$ in Algorithm 1, where L is defined in Equation (7.4). We evaluate our model's performance on all 4 datasets. On MNIST and Fashion-MNIST, Neural SFC model is trained for each class label. FFHQ has no labels, therefore all images are used together to learn a Neural SFC. On the TGIF dataset, Neural SFC model is trained on all gifs but generates a unique SFC for each gif, *i.e.*, all frames in a gif share the same Neural SFC.

The last column of Table 7.1 shows the average file size required to store an image in the test data and its relative improvement compared to the Raster scan. Specifically, in the TGIF section, the number represents the average size to save a single frame instead of the entire gif file. It is worth noting that this size does not include the order itself. One order can be shared

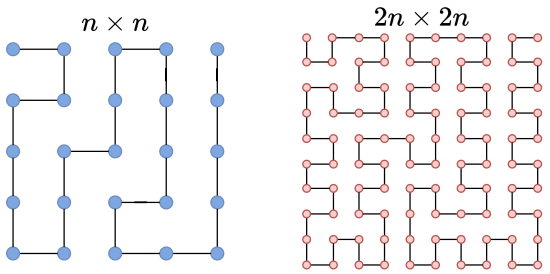
Table 7.1: Comparison of performance of lag-k autocorrelation and LZW Encoding length for different orders. For autocorrelation, we consistently outperform both the universal and context-based SFC computation approaches at high values of k. For LZW Encoding length, we measure the average size per frame in bytes as well as the relative improvement compared to the raster scan order, in the case of each of the datasets. We consistently outperform compression performance for other order schemes.

Dataset	Method	$\rho_6 \uparrow$	$\rho_{10} \uparrow$	Size in bytes (Δ) $\downarrow$
MNIST	Raster	0.206	0.102	175.4
	Hilbert	0.475	0.378	182.7 (+7.3)
	Dafner [306]	0.401	0.348	-
	Neural SFC (Ours)	**0.558**	**0.451**	**171.1 (-4.3)**
FMNIST	Raster	0.552	0.360	425.8
	Hilbert	0.7 23	0.647	427.3(+1.5)
	Dafner [306]	0.704	0.627	-
	Neural SFC (Ours)	**0.786**	**0.705**	**412.4 (-13.4)**
FFHQ	Raster	0.824	0.775	688.0
	Hilbert	0.924	0.899	689.6(+1.6)
	Dafner [306]	0.901	0.871	-
	Neural SFC (Ours)	**0.943**	**0.911**	**678.3 (-9.7)**
TGIF	Raster	-	-	563.9
	Hilbert	-	-	567.0 (+3.1)
	Neural SFC (Ours)	-	-	**556.9 (-7.0)**

by many images, thus the cost of it can be amortized. We observe that the Neural SFC results in smaller a sequence size on all 4 datasets, showing consistent improvement on the Raster scan or Hilbert Curve. Interestingly, as compared to the Raster scan, even though Hilbert Curve results in higher autocorrelation (see Figure 7.6), it is always worse in terms of LZW encoding length. This finding suggests that there is no simple relation between an order's autocorrelation performance and LZW encoding length performance.



(a) Scaling strategy for 2×2 grid



(b) SFC scaled from 5×5 resolution to 10×10 resolution

Figure 7.7: **Scaling up SFC.** The top row (blue circles) shows a 2×2 crop of an image grid of resolution $n \times n$. The bottom row (red circles) shows how we can scale the SFC path from $n \times n$ grid to $2n \times 2n$ grid. For every incoming SFC to a pixel location, there are 3 possible 'next' pixels, straight ahead, left, or right. Column 2 and 3 consider the cases where the SFC goes 'straight' or 'right'. The image on the right shows a toy example of scaling up SFC for a 5×5 image.

7.3.6 Scaling up SFC

Although in the current work, we only train Neural SFC models for 32×32 images to demonstrate their effectiveness, there are no theoretical restrictions of our approach for higher resolution images. It is trivial to either (1) train SFC for higher resolution itself or (2) scale an SFC computed for a low-resolution image to a high-resolution image, preserving the locality properties as shown in Fig. 7.7.

Chapter 8: Conclusion and Discussion

8.1 Summary

This dissertation proposes two quintessential building blocks for achieving compositionality in deep networks by design. The first block aims at discovering discrete primitives in images without supervision, which can be useful for low-shot downstream tasks such as classification, detection, retrieval, generation, and correspondence. The second block achieves the goal of combining the primitives in a data-driven and task-driven manner. We demonstrate the applications of this in content creation and compression.

In the first part of the thesis, we discussed two approaches for discovering discrete primitives in images. In Chapter 2, we formulated a way of representing images as a combination of a part appearance and part visibility maps where part appearances are as sampled from a codebook. These parts are consistent across the collection of images and can be used for image understanding as well as image editing applications. We further showed the application of these parts in the classification problem. In Chapter 3, we approached the problem of discovering primitives using a contrastive approach inspired by game theory. We play a cooperative referential game between two neural network agents, a speaker and a listener, who can communicate with each other only via a discrete bottleneck. This variable-length discrete bottleneck encodes images in the form of discrete latent primitives. We further showed the applications of this approach in classification,

detection, retrieval, and part segmentations. Finally in Chapter 4, we showed that it is possible to enhance these discrete primitives and learn a dense correspondence for a set of images.

In the second part of the thesis, we proposed two ideas for generating 2D and 3D shapes by composition. In Chapter 5, we use multiple camera viewpoints as individual primitives and synthesize a 3D object by generating depth maps from different viewpoints. These multiview depth maps can represent dense surface geometry and are typically not as heavyweight as their voxel or mesh counterparts. In Chapter 6, we use bounding boxes (with attributes) as our primitives, and developed LayoutTransformer, a transformer decoded based approach to autoregressively generate 2D and 3D scenes from scratch or from an initial seed set of primitives. Our method allows an arbitrary number of primitives per layout. It can work in both 2D and 3D domains, and can also have many direct and indirect applications in design, image, and 3D applications. Finally, in Chapter 7, we proposed a data-driven way to learn scan order of pixels in images. It addresses the limitations of LayoutTransfomer which can output primitives only in a raster scan order.

8.2 Discussion

While we proposed a couple of ways of discovering and composing primitives in images and 3D data, we have barely scratched the surface when it comes to solving the problem of compositional generalization. The methods we presented in the thesis can understand and generate compositions already seen during training with reasonable accuracy, however, in real-world applications such as robotics, we often need to *generalize beyond the seen compositions*. For example, a model that has learned to see/create a fish swimming during the training, may not be able to understand/synthesize a horse swimming because of a lack of training data for the latter.

Another major problem in compositional generalization is related to the *order of composition*. Composing multiple primitives is often not commutative which means the output may change if you compose things in a different order. There are tons of real-world examples where it is very important to apply primitives in a specific order. For instance, writing on a piece of paper and folding the paper is different from folding a piece of paper and then writing on it. Finally, handling a large number of primitives and understanding their relationships over long-time horizons are also some of the unsolved problems when it comes to 2D/3D video understanding and generation. While there have been a few attempts, especially in the reinforcement learning community, to try to imitate and align models with human understanding, yet we have a long way to go before we can have general-purpose models that can solve various vision applications in the open world.

There has been some exciting progress recently, especially with the text-to-image models, which with the help of a very large labeled training corpus, can generate realistic high-resolution images with text along. These models, however, still lack the 3D and compositional world understanding. While it might be possible to incorporate these with the assumptions of infinite data. But for practical scenarios, I hope this thesis highlights the importance of leveraging the ideas of compositionality while developing new approaches for visual recognition.

Bibliography

[1] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019.

[2] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. PointFlow: 3D point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4541–4550, 2019.

[3] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[4] Akash Abdu Jyothi, Thibaut Durand, Jiawei He, Leonid Sigal, and Greg Mori. Layoutvae: Stochastic scene layout generation from a label set. *arXiv preprint arXiv:1907.10719*, 2019.

[5] Bo Zhao, Lili Meng, Weidong Yin, and Leonid Sigal. Image generation from layout. In *CVPR*, 2019.

[6] Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1219–1228, 2018.

[7] Kamal Gupta, Alessandro Achille, Justin Lazarow, Larry Davis, Vijay Mahadevan, and Abhinav Shrivastava. LayoutTransformer: Layout Generation and Completion with Self-attention. In *ICCV*, 2020.

[8] 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. *ICLR*, 2020.

[9] Shir Amir, Yossi Gandelsman, Shai Bagon, and Tali Dekel. Deep vit features as dense visual descriptors. *arXiv preprint arXiv:2112.05814*, 2(3):4, 2021.

[10] Noam Chomsky. A minimalist program for linguistic theory. *MIT Press*, 1993.

[11] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40:e253, 2017.

[12] Kamal Gupta, Saurabh Singh, and Abhinav Shrivastava. Patchvae: Learning local latent codes for recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4746–4755, 2020.

[13] Kamal Gupta, Gowthami Somepalli, Anubhav Gupta, Jayasundara Vinoj, Matthias Zwicker, and Abhinav Shrivastava. PatchGame: Learning to Signal Mid-level Patches in Referential Games. In *NeurIPS*, 2021.

[14] **K. Gupta**, V. Jampani, C. Esteves, A. Shrivastava, A. Makadia, N. Snavely, and K. Kar. ASIC: Aligning sparse in-the-wild image collections. In *Under Review*, 2023.

[15] S. Girish, **K. Gupta**, Singh S., and A. Shrivastava. LilNetX - Lightweight Networks with EXtreme Model Compression and Structured Sparsification. In *ICLR*, 2022.

[16] S. Girish, SR. Maiya, **K. Gupta**, H. Chen, and A. Shrivastava. The Lottery Ticket Hypothesis for Object Recognition. In *CVPR*, 2021.

[17] H. Wang, **K. Gupta**, L. Davis, and A. Shrivastava. Neural Space-filling Curves. In *ECCV*, 2022.

[18] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 843–852, 2017.

[19] Linda Smith and Michael Gasser. The development of embodied cognition: Six lessons from babies. *Artificial life*, 11(1-2):13–29, 2005.

[20] Saurabh Singh, Abhinav Gupta, and Alexei A. Efros. Unsupervised discovery of mid-level discriminative patches. In *European Conference on Computer Vision*, 2012. URL `http://arxiv.org/abs/1205.3137`.

[21] Carl Doersch, Saurabh Singh, Abhinav Gupta, Josef Sivic, and Alexei A. Efros. What makes paris look like paris? *ACM Transactions on Graphics (SIGGRAPH)*, 31(4):101:1–101:9, 2012.

[22] Mayank Juneja, Andrea Vedaldi, CV Jawahar, and Andrew Zisserman. Blocks that shout: Distinctive parts for scene classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 923–930, 2013.

[23] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[24] Loïc Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR 2017*, 2017.

[25] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[26] Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 413–420. IEEE, 2009.

[27] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1452–1464, 2017.

[28] 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.

[29] Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pages 4743–4751, 2016.

[30] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

[31] Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *CoRR*, abs/1601.06759, 2016. URL `http://arxiv.org/abs/1601.06759`.

[32] Ishaan Gulrajani, Kundan Kumar, Faruk Ahmed, Adrien Ali Taiga, Francesco Visin, David Vazquez, and Aaron Courville. Pixelvae: A latent variable model for natural images. *arXiv preprint arXiv:1611.05013*, 2016.

[33] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. Pixelcnn++: A pixelcnn implementation with discretized logistic mixture likelihood and other modifications. In *ICLR*, 2017.

[34] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[35] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015. URL `http://arxiv.org/abs/1511.06434`.

[36] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2223–2232, 2017.

[37] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[38] Salman H Khan, Munawar Hayat, and Nick Barnes. Adversarial training of variational auto-encoders for high fidelity image generation. *arXiv preprint arXiv:1804.10323*, 2018.

[39] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

[40] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*, 2015.

[41] Lama Affara, Bernard Ghanem, and Peter Wonka. Supervised convolutional sparse coding. *CoRR*, abs/1804.02678, 2018. URL `http://arxiv.org/abs/1804.02678`.

[42] Marco Fraccaro, Simon Kamronn, Ulrich Paquet, and Ole Winther. A disentangled recognition and nonlinear dynamics model for unsupervised learning. In *NIPS*, 2017.

[43] Hyunjik Kim and Andriy Mnih. Disentangling by factorising. *arXiv preprint arXiv:1802.05983*, 2018.

[44] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, pages 3581–3589, 2014.

[45] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*, pages 3483–3491, 2015.

[46] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Context as supervisory signal: Discovering objects with predictable context. In *European Conference on Computer Vision*, pages 362–377. Springer, 2014.

[47] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. In *International Conference on Computer Vision (ICCV)*, 2015.

[48] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802, 2015.

[49] Deepak Pathak, Ross Girshick, Piotr Dollár, Trevor Darrell, and Bharath Hariharan. Learning features by watching objects move. In *CVPR*, 2017.

[50] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2536–2544, 2016.

[51] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer, 2016.

[52] Xiaolong Wang, Kaiming He, and Abhinav Gupta. Transitive invariance for self-supervised visual representation learning. In *Proc. of Int'l Conf. on Computer Vision (ICCV)*, 2017.

[53] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

[54] Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc V Gool. Soft-to-hard vector quantization for end-to-end learning compressible representations. In *Advances in Neural Information Processing Systems*, pages 1141–1151, 2017.

[55] 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.

[56] Steven Pinker. *The language instinct: How the mind creates language*. Penguin UK, 2003.

[57] Tomas Mikolov, Armand Joulin, and Marco Baroni. A roadmap towards machine intelligence. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 29–61. Springer, 2016.

[58] Herbert H Clark. *Using language*. Cambridge university press, 1996.

[59] Ranjay Krishna, Ines Chami, Michael Bernstein, and Li Fei-Fei. Referring relationships. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6867–6876, 2018.

[60] John Batali. Computational simulations of the emergence of grammar. *Approach to the Evolution of Language*, pages 405–426, 1998.

[61] Simon Kirby. Natural language from artificial life. *Artificial life*, 8(2):185–215, 2002.

[62] Luc Steels. What triggers the emergence of grammar? *Society for the Study of Artificial Intelligence and Simulation of Behaviour*, 2005.

[63] Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. Multi-agent cooperation and the emergence of (natural) language. *arXiv preprint arXiv:1612.07182*, 2016.

[64] Serhii Havrylov and Ivan Titov. Emergence of language with multi-agent games: Learning to communicate with sequences of symbols. *arXiv preprint arXiv:1705.11192*, 2017.

[65] Angeliki Lazaridou, Karl Moritz Hermann, Karl Tuyls, and Stephen Clark. Emergence of linguistic communication from referential games with symbolic and pixel input. *arXiv preprint arXiv:1804.03984*, 2018.

[66] Ludwig Wittgenstein. *Philosophical investigations*. John Wiley & Sons, 1953.

[67] David Lewis. *Convention: A philosophical study*. John Wiley & Sons, 1969.

[68] Frederic Jurie and Bill Triggs. Creating efficient codebooks for visual recognition. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 1, pages 604–610. IEEE, 2005.

[69] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[70] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[71] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. *arXiv preprint arXiv:1711.00937*, 2017.

[72] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[73] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *NeurIPS*, 2019.

[74] Saurabh Singh, Abhinav Gupta, and Alexei A Efros. Unsupervised discovery of mid-level discriminative patches. In *European Conference on Computer Vision*, pages 73–86. Springer, 2012.

[75] Carl Doersch, Saurabh Singh, Abhinav Gupta, Josef Sivic, and Alexei Efros. What makes paris look like paris? *ACM Transactions on Graphics*, 31(4), 2012.

[76] Katrina Evtimova, Andrew Drozdov, Douwe Kiela, and Kyunghyun Cho. Emergent communication in a multi-modal, multi-step referential game. *arXiv preprint arXiv:1705.10369*, 2017.

[77] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[78] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[79] Daniela Mihai and Jonathon Hare. The emergence of visual semantics through communication games. *arXiv preprint arXiv:2101.10253*, 2021.

[80] Kyle Wagner, James A Reggia, Juan Uriagereka, and Gerald S Wilkinson. Progress in the simulation of emergent communication and language. *Adaptive Behavior*, 11(1):37–69, 2003.

[81] Andrea Baronchelli, Maddalena Felici, Vittorio Loreto, Emanuele Caglioti, and Luc Steels. Sharp transition towards shared vocabularies in multi-agent systems. *Journal of Statistical Mechanics: Theory and Experiment*, 2006(06):P06014, 2006.

[82] Jacob Andreas, Dan Klein, and Sergey Levine. Learning with latent language. *arXiv preprint arXiv:1711.00482*, 2017.

[83] Jacob Andreas and Dan Klein. Analogs of linguistic structure in deep representations. *arXiv preprint arXiv:1707.08139*, 2017.

[84] Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. Referitgame: Referring to objects in photographs of natural scenes. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 787–798, 2014.

[85] Emilio Jorge, Mikael Kågebäck, Fredrik D Johansson, and Emil Gustavsson. Learning to play guess who? and inventing a grounded language as a consequence. *arXiv preprint arXiv:1611.03218*, 2016.

[86] Jakob N Foerster, Yannis M Assael, Nando De Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. *arXiv preprint arXiv:1605.06676*, 2016.

[87] Abhishek Das, Satwik Kottur, José MF Moura, Stefan Lee, and Dhruv Batra. Learning cooperative visual dialog agents with deep reinforcement learning. In *Proceedings of the IEEE international conference on computer vision*, pages 2951–2960, 2017.

[88] Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation. *arXiv preprint arXiv:1605.07736*, 2016.

[89] Jason Lee, Kyunghyun Cho, Jason Weston, and Douwe Kiela. Emergent translation in multi-agent communication. *arXiv preprint arXiv:1710.06922*, 2017.

[90] Matthew Spike, Kevin Stadler, Simon Kirby, and Kenny Smith. Minimal requirements for the emergence of learned signaling. *Cognitive science*, 41(3):623–658, 2017.

[91] Diane Bouchacourt and Marco Baroni. Miss tools and mr fruit: Emergent communication in agents learning about object affordances. *arXiv preprint arXiv:1905.11871*, 2019.

[92] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[93] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[94] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[95] Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[96] Ryan Lowe, Jakob Foerster, Y-Lan Boureau, Joelle Pineau, and Yann Dauphin. On the pitfalls of measuring emergent communication. *arXiv preprint arXiv:1903.05168*, 2019.

[97] Bence Keresztury and Elia Bruni. Compositional properties of emergent languages in deep learning. *arXiv preprint arXiv:2001.08618*, 2020.

[98] Jacob Andreas. Measuring compositionality in representation learning. *arXiv preprint arXiv:1902.07181*, 2019.

[99] Jason Tyler Rolfe. Discrete variational autoencoders. *arXiv preprint arXiv:1609.02200*, 2016.

[100] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *arXiv preprint arXiv:2102.12092*, 2021.

[101] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. *arXiv preprint arXiv:2012.09841*, 2020.

[102] Charlie Nash, Jacob Menick, Sander Dieleman, and Peter W Battaglia. Generating images with sparse representations. *arXiv preprint arXiv:2103.03841*, 2021.

[103] Kamal Gupta, Justin Lazarow, Alessandro Achille, Larry S Davis, Vijay Mahadevan, and Abhinav Shrivastava. Layouttransformer: Layout generation and completion with self-attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1004–1014, 2021.

[104] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

[105] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[106] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *arXiv preprint*, 2018.

[107] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *arXiv preprint arXiv:2104.14294*, 2021.

[108] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. *arXiv preprint arXiv:2103.03230*, 2021.

[109] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, pages 1597–1607. PMLR, 2020.

[110] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv preprint arXiv:2006.09882*, 2020.

[111] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. *arXiv preprint arXiv:2011.10566*, 2020.

[112] 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.

[113] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, pages 9729–9738, 2020.

[114] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

[115] Daniela Mihai and Jonathon Hare. Avoiding hashing and encouraging visual semantics in referential emergent language games. *arXiv preprint arXiv:1911.05546*, 2019.

[116] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 1857–1865, 2016.

[117] Emil Julius Gumbel. *Statistical theory of extreme values and some practical applications: a series of lectures*, volume 33. US Government Printing Office, 1954.

[118] Mathieu Blondel, Olivier Teboul, Quentin Berthet, and Josip Djolonga. Fast differentiable sorting and ranking. In *International Conference on Machine Learning*, pages 950–959. PMLR, 2020.

[119] 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.

[120] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[121] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, January 2015.

[122] Irving Biederman. Recognition-by-components: a theory of human image understanding. *Psychological review*, 94(2):115, 1987.

[123] Juhong Min, Jongmin Lee, Jean Ponce, and Minsu Cho. Hyperpixel flow: Semantic correspondence with multi-layer neural features. In *ICCV*, pages 3395–3404, 2019.

[124] Seokju Cho, Sunghwan Hong, Sangryul Jeon, Yunsung Lee, Kwanghoon Sohn, and Seungryong Kim. Cats: Cost aggregation transformers for visual correspondence. *NeurIPS*, 34:9011–9023, 2021.

[125] Nilesh Kulkarni, Abhinav Gupta, David F Fouhey, and Shubham Tulsiani. Articulation-aware canonical surface mapping. In *CVPR*, pages 452–461, 2020.

[126] Jason Zhang, Gengshan Yang, Shubham Tulsiani, and Deva Ramanan. Ners: Neural reflectance surfaces for sparse-view 3d reconstruction in the wild. *Advances in Neural Information Processing Systems*, 34:29835–29847, 2021.

[127] William Peebles, Jun-Yan Zhu, Richard Zhang, Antonio Torralba, Alexei A Efros, and Eli Shechtman. Gan-supervised dense visual alignment. In *CVPR*, pages 13470–13481, 2022.

[128] Jiteng Mu, Shalini De Mello, Zhiding Yu, Nuno Vasconcelos, Xiaolong Wang, Jan Kautz, and Sifei Liu. Coordgan: Self-supervised dense correspondences emerge from gans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10011–10020, 2022.

[129] Richard Szeliski et al. Image alignment and stitching: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 2(1):1–104, 2007.

[130] Erik G Learned-Miller. Data driven image models through continuous joint alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(2):236–250, 2005.

[131] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9446–9454, 2018.

[132] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[133] Yoni Kasten, Dolev Ofri, Oliver Wang, and Tali Dekel. Layered neural atlases for consistent video editing. *ACM Transactions on Graphics (TOG)*, 40(6):1–12, 2021.

[134] 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 *ICML*, pages 8748–8763. PMLR, 2021.

[135] Juhong Min, Jongmin Lee, Jean Ponce, and Minsu Cho. Spair-71k: A large-scale benchmark for semantic correspondence. *arXiv preprint arXiv:1908.10543*, 2019.

[136] Bumsub Ham, Minsu Cho, Cordelia Schmid, and Jean Ponce. Proposal flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

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

[138] Mark Boss, Andreas Engelhardt, Abhishek Kar, Yuanzhen Li, Deqing Sun, Jonathan T. Barron, Hendrik P.A. Lensch, and Varun Jampani. SAMURAI: Shape And Material from Unconstrained Real-world Arbitrary Image collections. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[139] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

[140] Michael J Black and Padmanabhan Anandan. A framework for the robust estimation of optical flow. In *1993 (4th) International Conference on Computer Vision*, pages 231–236. IEEE, 1993.

[141] Steven S. Beauchemin and John L. Barron. The computation of optical flow. *ACM computing surveys (CSUR)*, 27(3):433–466, 1995.

[142] G Lowe. Sift-the scale invariant feature transform. *Int. J*, 2(91-110):2, 2004.

[143] Bruce D Lucas, Takeo Kanade, et al. *An iterative image registration technique with an application to stereo vision*, volume 81. Vancouver, 1981.

[144] Carlo Tomasi and Takeo Kanade. Detection and tracking of point. *Int J Comput Vis*, 9: 137–154, 1991.

[145] Jan-Michael Frahm, Pierre Fite-Georgel, David Gallup, Tim Johnson, Rahul Raguram, Changchang Wu, Yi-Hung Jen, Enrique Dunn, Brian Clipp, Svetlana Lazebnik, et al. Building rome on a cloudless day. In *ECCV*, pages 368–381. Springer, 2010.

[146] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M Seitz, and Richard Szeliski. Building rome in a day. *Communications of the ACM*, 54(10): 105–112, 2011.

[147] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, pages 4104–4113, 2016.

[148] Ce Liu, Jenny Yuen, and Antonio Torralba. Sift flow: Dense correspondence across scenes and its applications. *IEEE TPAMI*, 33(5):978–994, 2010.

[149] Christopher B Choy, JunYoung Gwak, Silvio Savarese, and Manmohan Chandraker. Universal correspondence network. *Advances in neural information processing systems*, 29, 2016.

[150] Seungryong Kim, Dongbo Min, Bumsub Ham, Sangryul Jeon, Stephen Lin, and Kwanghoon Sohn. Fcss: Fully convolutional self-similarity for dense semantic correspondence. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6560–6569, 2017.

[151] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 224–236, 2018.

[152] Yanbin Liu, Linchao Zhu, Makoto Yamada, and Yi Yang. Semantic correspondence as an optimal transport problem. In *CVPR*, pages 4463–4472, 2020.

[153] Xin Li, Deng-Ping Fan, Fan Yang, Ao Luo, Hong Cheng, and Zicheng Liu. Probabilistic model distillation for semantic correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7505–7514, 2021.

[154] Wei Jiang, Eduard Trulls, Jan Hosang, Andrea Tagliasacchi, and Kwang Moo Yi. COTR: Correspondence transformer for matching across images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6207–6217, 2021.

[155] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. Loftr: Detector-free local feature matching with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8922–8931, 2021.

[156] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *CVPR*, pages 4938–4947, 2020.

[157] Ignacio Rocco, Mircea Cimpoi, Relja Arandjelović, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Neighbourhood consensus networks. *NeurIPS*, 31, 2018.

[158] Shuda Li, Kai Han, Theo W Costain, Henry Howard-Jenkins, and Victor Prisacariu. Correspondence networks with adaptive neighbourhood consensus. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10196–10205, 2020.

[159] Dongyang Zhao, Ziyang Song, Zhenghao Ji, Gangming Zhao, Weifeng Ge, and Yizhou Yu. Multi-scale matching networks for semantic correspondence. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3354–3364, 2021.

[160] Juhong Min and Minsu Cho. Convolutional hough matching networks. In *CVPR*, pages 2940–2950, 2021.

[161] Jae Yong Lee, Joseph DeGol, Victor Fragoso, and Sudipta N Sinha. Patchmatch-based neighborhood consensus for semantic correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13153–13163, 2021.

[162] Shuaiyi Huang, Luyu Yang, Bo He, Songyang Zhang, Xuming He, and Abhinav Shrivastava. Learning semantic correspondence with sparse annotations. *arXiv preprint arXiv:2208.06974*, 2022.

[163] Dariu M Gavrila. Multi-feature hierarchical template matching using distance transforms. In *Proceedings. Fourteenth international conference on pattern recognition (Cat. No. 98EX170)*, volume 1, pages 439–444. IEEE, 1998.

[164] Sergey Ioffe and David A. Forsyth. Probabilistic methods for finding people. *International Journal of Computer Vision*, 43(1):45–68, 2001.

[165] Gary B Huang, Vidit Jain, and Erik Learned-Miller. Unsupervised joint alignment of complex images. In *2007 IEEE 11th international conference on computer vision*, pages 1–8. IEEE, 2007.

[166] Gary Huang, Marwan Mattar, Honglak Lee, and Erik Learned-Miller. Learning to align from scratch. *Advances in neural information processing systems*, 25, 2012.

[167] 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.

[168] 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.

[169] Nilesh Kulkarni, Abhinav Gupta, and Shubham Tulsiani. Canonical surface mapping via geometric cycle consistency. *ICCV*, 2019.

[170] Angjoo Kanazawa, Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *ECCV*, 2018.

[171] Chun-Han Yao, Wei-Chih Hung, Yuanzhen Li, Michael Rubinstein, Ming-Hsuan Yang, and Varun Jampani. Lassie: Learning articulated shapes from sparse image ensemble via 3d part discovery. *arXiv preprint arXiv:2207.03434*, 2022.

[172] Paul Hongsuck Seo, Jongmin Lee, Deunsol Jung, Bohyung Han, and Minsu Cho. Attentive semantic alignment with offset-aware correlation kernels. In *ECCV*, pages 349–364, 2018.

[173] David Novotny, Samuel Albanie, Diane Larlus, and Andrea Vedaldi. Self-supervised learning of geometrically stable features through probabilistic introspection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3637–3645, 2018.

[174] Prune Truong, Martin Danelljan, Fisher Yu, and Luc Van Gool. Warp consistency for unsupervised learning of dense correspondences. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10346–10356, 2021.

[175] James Thewlis, Hakan Bilen, and Andrea Vedaldi. Unsupervised learning of object frames by dense equivariant image labelling. *Advances in neural information processing systems*, 30, 2017.

[176] Mehmet Aygün and Oisin Mac Aodha. Demystifying unsupervised semantic correspondence estimation. In *European Conference on Computer Vision*, pages 125–142. Springer, 2022.

[177] James Thewlis, Samuel Albanie, Hakan Bilen, and Andrea Vedaldi. Unsupervised learning of landmarks by descriptor vector exchange. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6361–6371, 2019.

[178] Sangryul Jeon, Seungryong Kim, Dongbo Min, and Kwanghoon Sohn. Parn: Pyramidal affine regression networks for dense semantic correspondence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 351–366, 2018.

[179] Ignacio Rocco, Relja Arandjelović, and Josef Sivic. End-to-end weakly-supervised semantic alignment. In *CVPR*, pages 6917–6925, 2018.

[180] Seungryong Kim, Dongbo Min, Somi Jeong, Sunok Kim, Sangryul Jeon, and Kwanghoon Sohn. Semantic attribute matching networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12339–12348, 2019.

[181] Sunghwan Hong and Seungryong Kim. Deep matching prior: Test-time optimization for dense correspondence. In *ICCV*, pages 9907–9917, 2021.

[182] Kfir Aberman, Jing Liao, Mingyi Shi, Dani Lischinski, Baoquan Chen, and Daniel Cohen-Or. Neural best-buddies: Sparse cross-domain correspondence. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018.

[183] Sangryul Jeon, Seungryong Kim, Dongbo Min, and Kwanghoon Sohn. Pyramidal semantic correspondence networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12):9102–9118, 2021.

[184] Dolev Ofri-Amar, Michal Geyer, Yoni Kasten, and Tali Dekel. Neural congealing: Aligning images to a joint semantic atlas. In *CVPR*, 2023.

[185] Subhabrata Choudhury, Iro Laina, Christian Rupprecht, and Andrea Vedaldi. Unsupervised part discovery from contrastive reconstruction. *Advances in Neural Information Processing Systems*, 34:28104–28118, 2021.

[186] Wei-Chih Hung, Varun Jampani, Sifei Liu, Pavlo Molchanov, Ming-Hsuan Yang, and Jan Kautz. Scops: Self-supervised co-part segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 869–878, 2019.

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

[188] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015.

[189] Jean Duchon. Splines minimizing rotation-invariant semi-norms in sobolev spaces. In *Constructive theory of functions of several variables*, pages 85–100. Springer, 1977.

[190] Peter J Huber. Robust estimation of a location parameter. *Breakthroughs in statistics: Methodology and distribution*, pages 492–518, 1992.

[191] 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.

[192] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. " grabcut" interactive foreground extraction using iterated graph cuts. *ACM transactions on graphics (TOG)*, 23(3): 309–314, 2004.

[193] Robert L Thorndike. Who belongs in the family. In *Psychometrika*. Citeseer, 1953.

[194] Scott Schaefer, Travis McPhail, and Joe Warren. Image deformation using moving least squares. In *ACM SIGGRAPH 2006 Papers*, pages 533–540, 2006.

[195] Ignacio Rocco, Relja Arandjelovic, and Josef Sivic. Convolutional neural network architecture for geometric matching. In *CVPR*, pages 6148–6157, 2017.

[196] Junghyup Lee, Dohyung Kim, Jean Ponce, and Bumsub Ham. Sfnet: Learning object-aware semantic correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2278–2287, 2019.

[197] 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.

[198] Andrew E Welchman. The human brain in depth: How we see in 3D. *Annual review of vision science*, 2:345–376, 2016.

[199] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *arXiv preprint arXiv:1812.04948*, 2018.

[200] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119, 2020.

[201] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan: How to embed images into the stylegan latent space? In *Proceedings of the IEEE international conference on computer vision*, pages 4432–4441, 2019.

[202] Lukas Fetty, Mikael Bylund, Peter Kuess, Gerd Heilemann, Tufve Nyholm, Dietmar Georg, and Tommy Löfstedt. Latent Space Manipulation for High-Resolution Medical Image Synthesis via the StyleGAN. *Zeitschrift für Medizinische Physik*, 2020.

[203] Weili Nie, Tero Karras, Animesh Garg, Shoubhik Debhath, Anjul Patney, Ankit B Patel, and Anima Anandkumar. Semi-Supervised StyleGAN for Disentanglement Learning. *arXiv*, pages arXiv–2003, 2020.

[204] Harold Scott Macdonald Coxeter. *Projective geometry*. Springer Science & Business Media, 1964.

[205] Shiyang Cheng, Michael Bronstein, Yuxiang Zhou, Irene Kotsia, Maja Pantic, and Stefanos Zafeiriou. MeshGAN: Non-linear 3D Morphable Models of Faces. *arXiv preprint arXiv:1903.10384*, 2019.

[206] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images. In *ECCV*, 2018.

[207] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. MeshCNN: A Network with an Edge. *ACM Transactions on Graphics (TOG)*, 38(4): 90, 2019.

[208] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017.

[209] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017.

[210] Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T Freeman, and Joshua B Tenenbaum. Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. In *Advances in Neural Information Processing Systems*, pages 82–90, 2016.

[211] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[212] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3D representations at high resolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3577–3586, 2017.

[213] Zhiqin Chen and Hao Zhang. Learning Implicit Fields for Generative Shape Modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[214] Shichen Liu, Shunsuke Saito, Weikai Chen, and Hao Li. Learning to Infer Implicit Surfaces without 3D Supervision. In *Advances in Neural Information Processing Systems 32*, pages 8295–8306. Curran Associates, Inc., 2019. URL `http://papers.nips.cc/paper/9039-learning-to-infer-implicit-surfaces-without-3d-supervision.pdf`.

[215] Yue Jiang, Dantong Ji, Zhizhong Han, and Matthias Zwicker. SDFdiff: Differentiable rendering of signed distance fields for 3D shape optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1251–1261, 2020.

[216] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. DISN: Deep Implicit Surface Network for High-quality Single-view 3D Reconstruction. In *Advances in Neural Information Processing Systems 32*, pages 492–502. Curran Associates, Inc., 2019. URL `http://papers.nips.cc/paper/8340-disn-deep-implicit-surface-network-for-high-quality-single-view-pdf`.

[217] Amir Arsalan Soltani, Haibin Huang, Jiajun Wu, Tejas D Kulkarni, and Joshua B Tenenbaum. Synthesizing 3D shapes via modeling multi-view depth maps and silhouettes with deep generative networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1511–1519, 2017.

[218] Siddhartha Chaudhuri, Evangelos Kalogerakis, Leonidas Guibas, and Vladlen Koltun. Probabilistic reasoning for assembly-based 3D modeling. In *ACM Transactions on Graphics (TOG)*, volume 30, page 35. ACM, 2011.

[219] Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, and Vladlen Koltun. A probabilistic model for component-based shape synthesis. In *ACM Transactions on Graphics (TOG)*, volume 31, page 55. ACM, 2012.

[220] Haibin Huang, Evangelos Kalogerakis, and Benjamin Marlin. Analysis and synthesis of 3D shape families via deep-learned generative models of surfaces. In *Computer Graphics Forum*, volume 34, pages 25–38. Wiley Online Library, 2015.

[221] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[222] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3D model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[223] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[224] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3D outputs. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2088–2096, 2017.

[225] Giorgos Bouritsas, Sergiy Bokhnyak, Michael Bronstein, and Stefanos Zafeiriou. Neural Morphable Models: Spiral Convolutional Networks for 3D Shape Representation Learning and Generation. *arXiv preprint arXiv:1905.02876*, 2019.

[226] Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J Black. Generating 3D faces using convolutional mesh autoencoders. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 704–720, 2018.

[227] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

[228] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan Russell, and Mathieu Aubry. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[229] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. 3d-coded: 3d correspondences by deep deformation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 230–246, 2018.

[230] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas J Guibas. Learning representations and generative models for 3D point clouds. *arXiv preprint arXiv:1707.02392*, 2017.

[231] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3D reconstruction in function space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019.

[232] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik G. Learned-Miller. Multi-view convolutional neural networks for 3D shape recognition. In *Proc. ICCV*, 2015.

[233] Alexey Dosovitskiy, Jost Tobias Springenberg, Maxim Tatarchenko, and Thomas Brox. Learning to generate chairs, tables and cars with convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(4):692–705, 2017.

[234] Samuel L. Smith, Pieter-Jan Kindermans, and Quoc V. Le. Don't decay the learning rate, increase the batch size. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=B1Yy1BxCZ`.

[235] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[236] Aäron van den Oord and Nal Kalchbrenner. Pixel RNN. *arXiv preprint*, 2016.

[237] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[238] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[239] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[240] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[241] Ke Li and Jitendra Malik. Implicit maximum likelihood estimation, 2019. URL `https://openreview.net/forum?id=rygunsAqYQ`.

[242] Yedid Hoshen, Ke Li, and Jitendra Malik. Non-adversarial image synthesis with generative latent nearest neighbors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5811–5819, 2019.

[243] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. URL `http://www.blender.org`.

[244] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. Shape completion using 3D-encoder-predictor cnns and shape synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5868–5877, 2017.

[245] Antonio Torralba and Pawan Sinha. Statistical context priming for object detection. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 1, pages 763–770. IEEE, 2001.

[246] Abhinav Shrivastava and Abhinav Gupta. Contextual priming and feedback for faster r-cnn. In *European Conference on Computer Vision*, pages 330–348. Springer, 2016.

[247] Irving Biederman. On the semantics of a glance at a scene. *Perceptual organization*, 213: 253, 1981.

[248] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

[249] 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.

[250] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[251] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*, 2018.

[252] Xiao Yang, Mehmet Ersin Yümer, Paul Asente, Mike Kraley, Daniel Kifer, and C. Lee Giles. Learning to extract semantic structure from documents using multimodal fully convolutional neural network. *CoRR*, abs/1706.02337, 2017. URL `http://arxiv.org/abs/1706.02337`.

[253] Samuele Capobianco and Simone Marinai. Docemul: a toolkit to generate structured historical documents. *CoRR*, abs/1710.03474, 2017. URL `http://arxiv.org/abs/1710.03474`.

[254] Angel X. Chang, Will Monroe, Manolis Savva, Christopher Potts, and Christopher D. Manning. Text to 3d scene generation with rich lexical grounding. *CoRR*, abs/1505.06289, 2015. URL `http://arxiv.org/abs/1505.06289`.

[255] Jiajun Wu, Joshua B Tenenbaum, and Pushmeet Kohli. Neural scene de-rendering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[256] Jiajun Wu, Erika Lu, Pushmeet Kohli, William T Freeman, and Joshua B Tenenbaum. Learning to see physics via visual de-animation. In *Advances in Neural Information Processing Systems*, 2017.

[257] Jianan Li, Jimei Yang, Aaron Hertzmann, Jianming Zhang, and Tingfa Xu. Layoutgan: Generating graphic layouts with wireframe discriminators. *arXiv preprint arXiv:1901.06767*, 2019.

[258] Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. Rico: A mobile app dataset for building data-driven design applications. In *Proceedings of the 30th Annual Symposium on User Interface Software and Technology*, UIST '17, 2017.

[259] Xu Zhong, Jianbin Tang, and Antonio Jimeno Yepes. Publaynet: largest dataset ever for document layout analysis. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1015–1022. IEEE, 2019.

[260] 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 *European conference on computer vision*, pages 740–755. Springer, 2014.

[261] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013.

[262] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *International Conference on Machine Learning*, pages 1691–1703. PMLR, 2020.

[263] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. *ICLR*, 2015.

[264] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *Advances in neural information processing systems*, pages 613–621, 2016.

[265] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in neural information processing systems*, pages 82–90, 2016.

[266] Kamal Gupta, Susmija Jabbireddy, Ketul Shah, Abhinav Shrivastava, and Matthias Zwicker. Improved modeling of 3d shapes with multi-view depth maps. *arXiv preprint arXiv:2009.03298*, 2020.

[267] Wenbo Li, Pengchuan Zhang, Lei Zhang, Qiuyuan Huang, Xiaodong He, Siwei Lyu, and Jianfeng Gao. Object-driven text-to-image synthesis via adversarial training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12174–12182, 2019.

[268] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5907–5915, 2017.

[269] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396*, 2016.

[270] Boren Li, Boyu Zhuang, Mingyang Li, and Jian Gu. Seq-sg2sl: Inferring semantic layout from scene graph through sequence to sequence learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7435–7443, 2019.

[271] Oron Ashual and Lior Wolf. Specifying object attributes and relations in interactive scene generation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4561–4569, 2019.

[272] Hao Dong, Simiao Yu, Chao Wu, and Yike Guo. Semantic image synthesis via adversarial learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5706–5714, 2017.

[273] Tobias Hinz, Stefan Heinrich, and Stefan Wermter. Generating multiple objects at spatially distinct locations. *CoRR*, abs/1901.00686, 2019. URL `http://arxiv.org/abs/1901.00686`.

[274] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arxiv*, 2016.

[275] 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*, 2018.

[276] Donghoon Lee, Sifei Liu, Jinwei Gu, Ming-Yu Liu, Ming-Hsuan Yang, and Jan Kautz. Context-aware synthesis and placement of object instances. *CoRR*, abs/1812.02350, 2018. URL `http://arxiv.org/abs/1812.02350`.

[277] Kai Wang, Manolis Savva, Angel X Chang, and Daniel Ritchie. Deep convolutional priors for indoor scene synthesis. *ACM Transactions on Graphics (TOG)*, 37(4):70, 2018.

[278] Kai Wang, Yu-An Lin, Ben Weissmann, Manolis Savva, Angel X Chang, and Daniel Ritchie. Planit: Planning and instantiating indoor scenes with relation graph and spatial prior networks. *ACM Transactions on Graphics (TOG)*, 38(4):1–15, 2019.

[279] Manyi Li, Akshay Gadi Patil, Kai Xu, Siddhartha Chaudhuri, Owais Khan, Ariel Shamir, Changhe Tu, Baoquan Chen, Daniel Cohen-Or, and Hao Zhang. Grains: Generative recursive autoencoders for indoor scenes. *ACM Transactions on Graphics (TOG)*, 38(2):1–16, 2019.

[280] Daniel Ritchie, Kai Wang, and Yu-an Lin. Fast and flexible indoor scene synthesis via deep convolutional generative models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6182–6190, 2019.

[281] Xinru Zheng, Xiaotian Qiao, Ying Cao, and Rynson WH Lau. Content-aware generative modeling of graphic design layouts. *ACM Transactions on Graphics (TOG)*, 38(4):1–15, 2019.

[282] Akshay Gadi Patil, Omri Ben-Eliezer, Or Perel, Hadar Averbuch-Elor, and Cornell Tech. Read: Recursive autoencoders for document layout generation. *arXiv preprint arXiv:1909.00302*, 2019.

[283] Dipu Manandhar, Dan Ruta, and John Collomosse. Learning structural similarity of user interface layouts using graph networks. In *ECCV*, 2020.

[284] Chuhang Zou, Ersin Yumer, Jimei Yang, Duygu Ceylan, and Derek Hoiem. 3d-prnn: Generating shape primitives with recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 900–909, 2017.

[285] Rundi Wu, Yixin Zhuang, Kai Xu, Hao Zhang, and Baoquan Chen. Pq-net: A generative part seq2seq network for 3d shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 829–838, 2020.

[286] Minhyuk Sung, Hao Su, Vladimir G Kim, Siddhartha Chaudhuri, and Leonidas Guibas. Complementme: weakly-supervised component suggestions for 3d modeling. *ACM Transactions on Graphics (TOG)*, 36(6):1–12, 2017.

[287] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications, 2017. *arXiv preprint arXiv:1701.05517*, 2017.

[288] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[289] Fenggen Yu, Kun Liu, Yan Zhang, Chenyang Zhu, and Kai Xu. Partnet: A recursive part decomposition network for fine-grained and hierarchical shape segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9491–9500, 2019.

[290] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019.

[291] Seunghoon Hong, Dingdong Yang, Jongwook Choi, and Honglak Lee. Inferring semantic layout for hierarchical text-to-image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7986–7994, 2018.

[292] Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

[293] Volker Steinbiss, Bach-Hiep Tran, and Hermann Ney. Improvements in beam search. In *Third International Conference on Spoken Language Processing*, 1994.

[294] Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy Mitra, and Leonidas J Guibas. Structurenet: Hierarchical graph networks for 3d shape generation. *arXiv preprint arXiv:1908.00575*, 2019.

[295] Tanmay Gupta, Alexander Schwing, and Derek Hoiem. Vico: Word embeddings from visual co-occurrences. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7425–7434, 2019.

[296] Thomas F. Liu, Mark Craft, Jason Situ, Ersin Yumer, Radomir Mech, and Ranjitha Kumar. Learning design semantics for mobile apps. In *The 31st Annual ACM Symposium on User Interface Software and Technology*, UIST '18, pages 569–579, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5948-1. doi: 10.1145/3242587.3242650. URL `http://doi.acm.org/10.1145/3242587.3242650`.

[297] David Hilbert. Über die stetige abbildung einer linie auf ein flächenstück. In *Dritter Band: Analysis· Grundlagen der Mathematik· Physik Verschiedenes*, pages 1–2. Springer, 1935.

[298] Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[299] Ian H Witten, Radford M Neal, and John G Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, 1987.

[300] David A Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[301] Giuseppe Peano. Sur une courbe, qui remplit toute une aire plane. *Mathematische Annalen*, 36(1):157–160, 1890.

[302] Bongki Moon, Hosagrahar V Jagadish, Christos Faloutsos, and Joel H. Saltz. Analysis of the clustering properties of the hilbert space-filling curve. *IEEE Transactions on knowledge and data engineering*, 13(1):124–141, 2001.

[303] Seiichiro Kamata, Richard O Eason, and Eiji Kawaguchi. An implementation of the hilbert scanning algorithm and its application to data compression. *IEICE TRANSACTIONS on Information and Systems*, 76(4):420–428, 1993.

[304] A Ansari and A Fineberg. Image data compression and ordering using peano scan and lot. *IEEE Trans. on Consumer Electronics*, 38(3):436–445, 1992.

[305] KS Thyagarajan and S Chatterjee. Fractal scanning for image compression. In *Conference Record of the Twenty-Fifth Asilomar Conference on Signals, Systems & Computers*, pages 467–468. IEEE Computer Society, 1991.

[306] Revital Dafner, Daniel Cohen-Or, and Yossi Matias. Context-based space filling curves. In *Computer Graphics Forum*, volume 19, pages 209–218. Wiley Online Library, 2000.

[307] Wacław Sierpínski. Sur une nouvelle courbe continue qui remplit toute une aire plane. *Bull. Acad. Sci. Cracovie (Sci. math. et nat. Serie A)*, pages 462–478, 1912.

[308] Eliakim Hastings Moore. On certain crinkly curves. *Transactions of the American Mathematical Society*, 1(1):72–90, 1900.

[309] Abraham Lempel and Jacob Ziv. Compression of two-dimensional data. *IEEE transactions on information theory*, 32(1):2–8, 1986.

[310] Michael Bader. *Space-filling curves: an introduction with applications in scientific computing*, volume 9. Springer Science & Business Media, 2012.

[311] VV Alexandrov, AI Alexeev, and ND Gorsky. A recursive algorithm for pattern recognition. *Proc. IEEE Intl. Conf. Pattern Recognition*, pages 431–433, 1982.

[312] Jia-Hong Lee and Yuang-Cheh Hsueh. Texture classification method using multiple space filling curves. *Pattern Recognition Letters*, 15(12):1241–1244, 1994.

[313] Yossi Matias and Adi Shamir. A video scrambling technique based on space filling curves. In *Conference on the theory and application of cryptographic techniques*, pages 398–417. Springer, 1987.

[314] Jonathan K Lawder. Calculation of mappings between one and n-dimensional values using the hilbert space-filling curve. *School of Computer Science and Information Systems, Birkbeck College, University of London, London Research Report BBKCS-00-01 August*, 2000.

[315] Jinhui Zhu, Ahmad Hoorfar, and Nader Engheta. Bandwidth, cross-polarization, and feed-point characteristics of matched hilbert antennas. *IEEE Antennas and Wireless Propagation Letters*, 2:2–5, 2003.

[316] Erez Lieberman-Aiden, Nynke L Van Berkum, Louise Williams, Maxim Imakaev, Tobias Ragoczy, Agnes Telling, Ido Amit, Bryan R Lajoie, Peter J Sabo, Michael O Dorschner, et al. Comprehensive mapping of long-range interactions reveals folding principles of the human genome. *science*, 326(5950):289–293, 2009.

[317] Randall Munroe. xkcd: Map of the internet. `https://xkcd.com/195`, 2006-12-11. Accessed: 2021-11-16.

[318] Maciej Drozdowski. *Scheduling for parallel processing*, volume 18. Springer, 2009.

[319] Andy B Yoo, Morris A Jette, and Mark Grondona. Slurm: Simple linux utility for resource management. In *Workshop on job scheduling strategies for parallel processing*, pages 44–60. Springer, 2003.

[320] Tarek Ouni, Arij Lassoued, and Mohamed Abid. Gradient-based space filling curves: Application to lossless image compression. In *2011 IEEE International Conference on Computer Applications and Industrial Electronics (ICCAIE)*, pages 437–442. IEEE, 2011.

[321] Liang Zhou, Chris R Johnson, and Daniel Weiskopf. Data-driven space-filling curves. *IEEE transactions on visualization and computer graphics*, 27(2):1591–1600, 2020.

[322] John J Hopfield and David W Tank. "neural" computation of decisions in optimization problems. *Biological cybernetics*, 52(3):141–152, 1985.

[323] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. *NIPS*, 2015.

[324] Michel Deudon, Pierre Cournut, Alexandre Lacoste, Yossiri Adulyasak, and Louis-Martin Rousseau. Learning Heuristics for the TSP by Policy Gradient. In *International conference on the integration of constraint programming, artificial intelligence, and operations research*, pages 170–181. Springer, 2018.

[325] Hanjun Dai, Elias B Khalil, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. *NIPS*, 2017.

[326] Wouter Kool, Herke Van Hoof, and Max Welling. Attention, learn to solve routing problems! *ICLR*, 2019.

[327] Frank Harary and Robert Z Norman. Some properties of line digraphs. *Rendiconti del Circolo Matematico di Palermo*, 9(2):161–168, 1960.

[328] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *ICLR*, 2016.

[329] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rJXMpikCZ`.

[330] Jacob Ziv and Abraham Lempel. Compression of individual sequences via variable-rate coding. *IEEE transactions on Information Theory*, 24(5):530–536, 1978.

[331] Terry A. Welch. Technique for high-performance data compression. *Computer*, 1984.

[332] Robert Clay Prim. Shortest connection networks and some generalizations. *The Bell System Technical Journal*, 36(6):1389–1401, 1957.

[333] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[334] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[335] 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.

[336] Yuncheng Li, Yale Song, Liangliang Cao, Joel Tetreault, Larry Goldberg, Alejandro Jaimes, and Jiebo Luo. Tgif: A new dataset and benchmark on animated gif description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4641–4650, 2016.