ABSTRACT

Title of Dissertation: Assimilating and Assembling Primitives for the Visual World

Kamal Gupta, 2021

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

Professor Larry S. Davis
Department of Computer Science
University of Maryland, College Park

In the last few years, there have been a number of works demonstrating the use of deep generative models to create visual data. These models are able to convert sketches into paintings, transfer style from one image to another, or convert an image of semantic labels to realistic scenes, and vice versa. Without a doubt, computational approaches will play a pivotal role in revolutionizing the way we create visual data. Existing approaches in unsupervised or generative modeling, often perform image synthesis in one shot using a black-box model. Arguably humans, on the other hand, follow two quintessential steps in the process of creation - assimilation, and assembly. In this dissertation, we take a deeper look inside each of the two steps and lay the groundwork on how deep generative models can be repurposed to aid humans with each of these steps.

The first part of the thesis focuses on "assimilation," the process of consuming the data and understanding various constituent components. We draw inspiration from works on the mid-level representation of images that aim to discover concepts in the form of patches that may correspond to objects or parts of objects. In this context, we present - PatchVAE and PatchGame, two complementary approaches to discover such discrete concepts in the image

which occur repetitively across the dataset and also compose in different ways to form an image.

The second part of the thesis focuses on "assembly," the process of synthesizing a meaningful arrangement of some known concepts to give rise to novel scenes sampled from the desired data distribution. Towards this, we first present building blocks for assembling different modalities of visual data: assembling multiple views (Multiview Shapes) and assembling graphical primitives (LayoutTransformer). Lastly, we take a sneak peek into our current work on synthesizing textured meshes and never-ending scenes.

Assimilating and Assembling 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
2021

Advisory Committee:
Professor Abhinav Shrivastava, Advisor
Professor Larry S. Davis, Co-Adviser
Professor David Jacobs
Professor Matthias Zwicker, Department Representative
Professor Noah Snavely, Cornell and Google

Table of contents

Chapter 1

Introduction

Every good composition is above all a work of abstraction

Diego Rivera

What I cannot create, I do not understand

Richard Feynman

For centuries, art and technology have intertwined with each other to allow creatives such as scientists, painters, or poets to distill knowledge from nature, people, culture, histories, and transform their ideas into reality. From stone tools to charcoal drawings, oil paintings to digital photography, technology has provided artists with new forms of expression since its inception. Many of these technologies in the past have often met apprehension and fear that they would render artists irrelevant, however time and again we have learned that these tools end up giving the artists new expressive opportunities.

We are living in times where we are witnessing the rise of another technology transforming the ways visual content is created. Several computational approaches led by deep neural networks, from deep-dream to neural style transfer, are allowing the creation of amazing new visual arts. With open-source repositories, data, and social media, access to these tools is easier than ever. Without a doubt, software built upon these approaches will play a pivotal role in enabling storytellers to create visual content in the future. The storytellers here, are not just designers and artists, but also the educators trying to create visual content for teaching, the architects designing cities for the next billion, entertainers live-streaming 3D movies or live-action sports to our homes, and the scientists and engineers collaborating from all over the world to bring their ideas to life. Be it VFX, games, advertising, video, medicine, art, or synthetic data for training, when it comes to representing and recreating the visual world, it is needless to say that the possibilities are endless.

Of course, numerous challenges need to be addressed. To begin with, while we live in a world of three dimensions, one of the most common representations of the world, an image,

is two-dimensional. Convolutional neural networks (CNNs) have been explicitly designed to exploit the two-dimensional structure of images and have achieved major successes in the last decade in several computer vision tasks. State-of-the-art works [Kar+17; KLA19; Par+19b; Iso+16; OK16; Sal+17c; Sal+17b] can generate highly realistic images of various image categories, such as human faces, cars, bedrooms, and many of the ImageNet classes.

However when it comes to 3D data representation, there are no gold standards. Based on the application, the representations vary widely from multi-view images, depth maps, point clouds, meshes, implicit surfaces, or voxels, and need to optimize for both the amount of memory consumed, and the compute required for processing and visualization. Some objects can be modeled by the surface geometry alone, while some need to be modeled by complete volumes. As we move away from simple 3D objects to deformable/translucent/luminous objects, 3D scenes, and videos, the modeling challenges grow. Apart from reconstruction and visualization, objects need to obey certain physical laws, generative models should not only ensure realism but also diversity, given a long-tail distribution of objects in the real world. Many applications resort to using very domain-specific knowledge to represent 3D objects, which vary drastically based on the properties of the domain.

The focus of this thesis is on building systems that can understand and create the visual content with minimum reliance on domain-specific knowledge. Our visual world, although incredibly complex, is also highly structured. This structure often appears in the form of a few primitives repeating in a very large number of configurations and is what allows for machine learning to be possible. Biederman [Bie81] provided strong evidence in cognitive neuroscience that perceiving and understanding a scene involves two related processes - (1) perceiving the visual signal and understanding its various constituent components, (2) modeling the relationships between these elements. Driven by this hypothesis, the majority of my research involves abstracting and solving these two challenges in perception and modeling of the 3D visual world.

1.1 Discovering Primitives for Complex Scenes

Compositionality and modularity is a core characteristic of representations designed by humans. In language, for instance, simple primitives such as alphabets can combine to make words which can further generate infinitely many sentences with complex meanings. Real world scenes and objects can also be decomposed into simple parts that are related to each other by various relationships or symmetries. In order to better understand the guiding principles behind generative process of data, it is imperative to build representations that can discover these simple parts or primitives. In images, representations learned by popular unsupervised and self-supervised machine learning systems, however follow a distributed paradigm. The question of whether distributed representations such as the ones learned by neural networks, can exhibit compositional structure has been long debated in classical

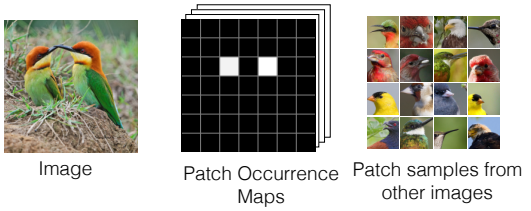


Figure 1.1: Given an image, we propose to learns mid-level patches in the image which occur repetitively across the data.

connectionist-symbolicist debates [Smo88]. How well are these representations able to capture high-level compositional primitives in images has been scarcely studied.

In Chapters 2 and 3, we propose computational models and learning algorithms to capture discrete primitives that appear in images, and can be composed to either reconstruct the image, or to differentiate the image from other images in the dataset.

1.2 Assembling Primitives to Model Complex Scenes

Conventional ML tools excel at processing signals such as images, however they can not be trivially applied to model 3D geometric data. In order to build tools to help artists create 3D visual content - we propose to work directly in the 3D models space. In Chapter 4, we propose an approach to directly generate 3D shapes from scratch or in an autoregressive way which allows a model to take as input, partial shape designed by an artist, and propose multiple possible completions of the shape. Further in Chapter 5, we suggest a way to texture the 3D shapes. The inspiration of the two steps comes from the works on Game Level Design, each of which are time-consuming and expensive process since they can only be accomplished by a team of professional artists. Our work will allow the artists to iterate fast while creating 3D scenes.

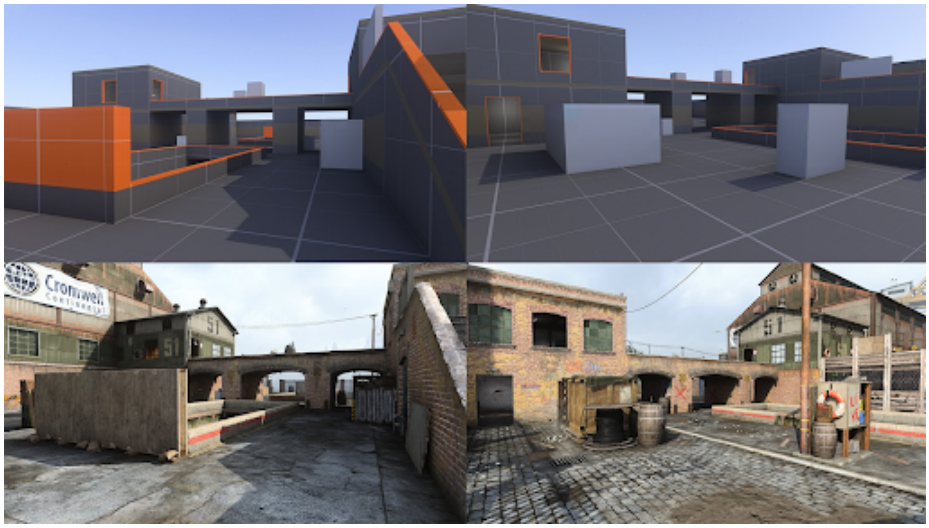


Figure 1.2: We take inspirations from works on Game Level Design []. A game level designer starts with a concept of the game, and draws a blockmesh (top row). A blockmesh is a blueprint of the game. Another team of artists texture the blockmesh to final scene (bottom row).

Chapter 2

PatchVAE: Learning Local Latent Codes for Recognition

2.1 Introduction

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 [Sun+17a]. 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 [SG05]. 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, 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

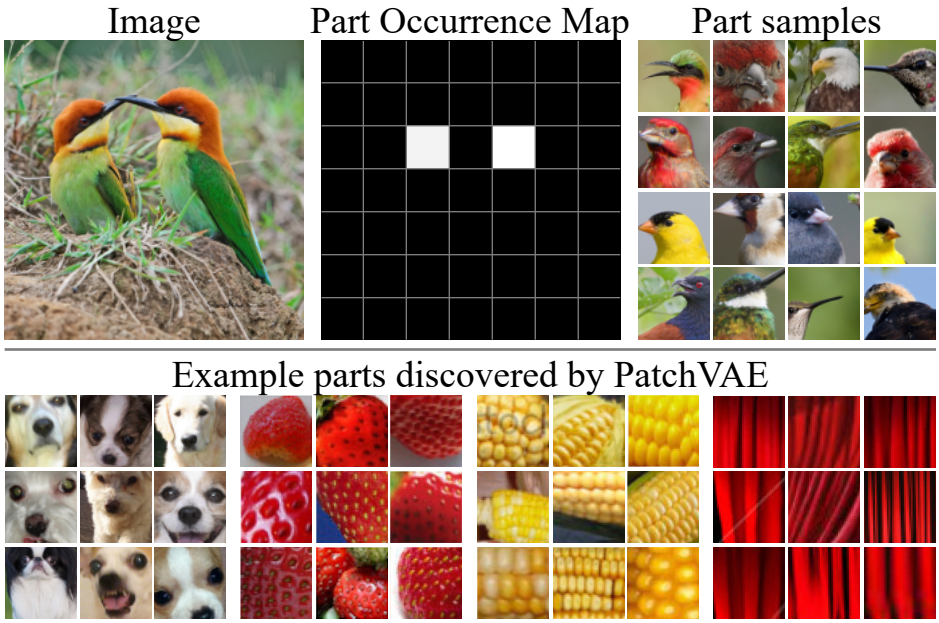


Figure 2.1: PatchVAE learns to encode repetitive parts across a dataset, by modeling their appearance and occurrence. (top) Given an image, the occurrence map of a particular part learned by PatchVAE is shown in the middle, capturing the head/beak of the birds. Samples of the same part from other images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

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 [SGE12b; Doe+12a; Jun+13]. 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) [KW13b; Mat+17], 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 4.1 shows an example of the discrete latents or occurrence map, and example parts discovered by our approach, PatchVAE. We present PatchVAE in Section 4.3 and

demonstrate that it learns representations that are much better for recognition as compared to those learned by the standard β-VAEs [KW13b; Mat+17].

In addition, we present losses that favor foreground, which is more likely to contain repetitive patterns, in Section 2.3.4, and demonstrate that they result in representations that are much better at recognition. Finally, in Section 2.4, we present results on CIFAR100 [KH+09], MIT Indoor Scene Recognition [QT09], Places [Zho+17], and ImageNet [Den+09] datasets. To summarize, our contributions are as follows:

1. We propose a novel patch-based bottleneck in the VAE framework that learns representations that can encode repetitive parts across images.

2. We demonstrate that our method, PatchVAE, learns unsupervised representations that are better suited for recognition in comparison to traditional VAEs.

3. We show that losses that favor foreground are better for unsupervised representation learning for recognition.

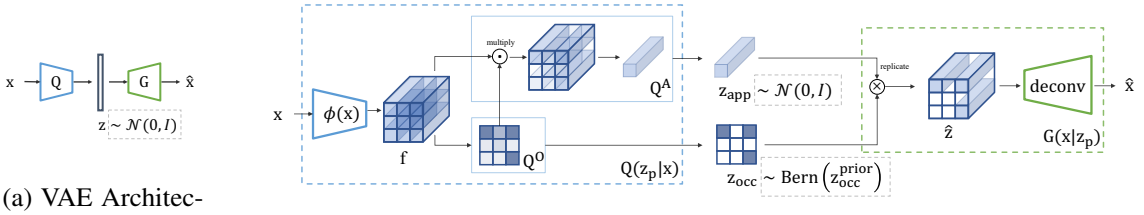
4. We perform extensive ablation analysis of the proposed PatchVAE architecture.

2.2 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 [KW13b; Mat+17; Kin+16; Gre+15], PixelRNN [OKK16], PixelCNN [Gul+16; Sal+17a], 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 [Goo+14; RMC15; Zhu+17; ACB17] 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 [KHB18; DKD16; Lar+15]. Convolution sparse coding [AGW18] 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 [Mat+17; Fra+17; KM18]. However, there is little evidence of effectiveness of disentangled representations in visual recognition. Semi-supervised learning using generative models [Kin+14; SLY15], 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



(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 $\mathbf{z}_{\text{occ}}$ and $\mathbf{z}_{\text{app}}$ to construct $\hat{\mathbf{z}}$ as described in Section 2.3.2 which is input to the decoder network. We also visualize the corresponding priors for latents $\mathbf{z}_{\text{app}}$ and $\mathbf{z}_{\text{occ}}$ in the dashed gray boxes.

tasks like arranging patches in an image in the correct spatial order [DGE14; DGE15] and arranging frames from a video in correct temporal order [WG15; Pat+17], to more involved tasks like in-painting [Pat+16] and context prediction [NF16; WHG17]. We follow the best practices from this line of work for evaluating the learned representations.

2.3 Our Approach

Our work builds upon VAE framework proposed by [KW13b]. We briefly review relevant aspects of the VAE framework and then present our approach.

2.3.1 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 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. Fig. 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 Eq. 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 [KW13b]. [Mat+17] 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 $\mathbf{z}$ represent 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 [SGE12b; Doe+12a; Jun+13], 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 2.3.2, we describe our approach in detail and in Section 2.3.4 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 [SGE12b]).

2.3.2 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 2.3.3).

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, $Q_{\mathbf{f}}^{\mathrm{A}}$ and $Q_{\mathbf{f}}^{\mathrm{O}}$, to parameterize posterior distributions $Q_{\mathbf{f}}^{\mathrm{A}}(\mathbf{z}_{\text{app}} \mid \mathbf{f})$ and $Q_{\mathbf{f}}^{\mathrm{O}}(\mathbf{z}_{\text{occ}}^{l} \mid \mathbf{f})$ of the part parameters $\mathbf{z}_{\text{app}}$ and $\mathbf{z}_{\text{occ}}^{l}$ respectively. Since the mapping $\mathbf{f} = \phi(\mathbf{x})$ is deterministic, we can re-write these distributions as $Q_{\mathbf{f}}^{\mathrm{A}}(\mathbf{z}_{\text{app}} \mid \phi(\mathbf{x}))$ and $Q_{\mathbf{f}}^{\mathrm{O}}(\mathbf{z}_{\text{occ}}^{l} \mid \phi(\mathbf{x}))$; or simply $Q^{\mathrm{A}}(\mathbf{z}_{\text{app}} \mid \mathbf{x})$ and $Q^{\mathrm{O}}(\mathbf{z}_{\text{occ}}^{l} \mid \mathbf{x})$. Therefore, given an image $\mathbf{x}$ the encoder networks estimate the posterior $Q^{\mathrm{A}}(\mathbf{z}_{\text{app}} \mid \mathbf{x})$ and $Q^{\mathrm{O}}(\mathbf{z}_{\text{occ}}^{l} \mid \mathbf{x})$. Note that $\mathbf{f}$ is a deterministic feature map, whereas $\mathbf{z}_{\text{app}}$ and $\mathbf{z}_{\text{occ}}^{l}$ 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 $\hat{\mathbf{z}}_{\text{occ}}^{l}$ (L dimensional, binary) one for each location l from the posteriors

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

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., $\hat{\mathbf{z}}_{\text{occ}}^{l} = 1$). This can be implemented by a broadcasted product of $\hat{\mathbf{z}}_{\text{app}}$ and $\hat{\mathbf{z}}_{\text{occ}}^{l}$. 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

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

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

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

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

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

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

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

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

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

Implementation details. As discussed in Section 2.3.1, the first and second terms of the RHS of (2.8) can be trained using L2 reconstruction loss and reparameterization trick [KW13b]. 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 [MMT16; Agu+17] for training part occurrence distributions $\mathbf{z}_{\text{occ}}^{l}$.

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 $Q_{\mathbf{f}}^{\mathrm{O}}(\mathbf{z}_{\text{occ}}^{l} \mid \mathbf{f})$ are single layer neural networks to compute $\mathbf{z}_{\text{app}}$ and $\mathbf{z}_{\text{occ}}^{l}$. $\mathbf{z}_{\text{occ}}^{l}$ 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 $\mathbf{z}_{\text{occ}}^{l}$ 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 $\sum_l \mathbf{z}_{\text{occ}}^{l} > 1$, 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 3.1). During image generation, we sample $\hat{\mathbf{z}}_{\text{app}}$ once and replicate it at each location where $\hat{\mathbf{z}}_{\text{occ}}^{l} = 1$. During training, this forces the model to: (1) only predict $\hat{\mathbf{z}}_{\text{occ}}^{l} = 1$ 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.

2.3.3 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 $Q^{\mathrm{A}(i)}\left(\mathbf{z}_{\text{app}}^{(i)} \mid \mathbf{x}\right)$ and $Q^{\mathrm{O}(i)}\left(\mathbf{z}_{\text{occ}}^{l(i)} \mid \mathbf{x}\right)$, where $\mathbf{z}_{\text{app}}^{(i)}$ and $\mathbf{z}_{\text{occ}}^{l(i)}$ parameterize 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, (2.6) can be written as:

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

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

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

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

2.3.4 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 ($\mathcal{L}_{\mathrm{w}}$)**. Note that this is similar to the gradient-energy biased sampling of mid-level patches used in [SGE12b; Doe+12a]. Examples of weight masks are provided in the supplemental material.

In addition, we also consider an adversarial training strategy from GANs to train VAEs [Lar+15], 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 2.4.2, we demonstrate that the proposed weighted loss ($\mathcal{L}_{\mathrm{w}}$) is complementary to the discriminator loss from adversarial training, and these losses result in better recognition capabilities for both β-VAE and PatchVAE.

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 2.4.1

Model	CIFAR100			Indoor67			Places205		
	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

2.4 Experiments

Datasets. We evaluate PatchVAE on CIFAR100 [KH+09], MIT Indoor Scene Recognition [QT09], Places [Zho+17] and Imagenet [Den+09] 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 [He+16]. 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

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 2.4.1

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

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 [Agu+17]). 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.

Baselines. We use the **β-VAE** model (Section 2.3.1) 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 [DKD16]. 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.

2.4.1 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 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

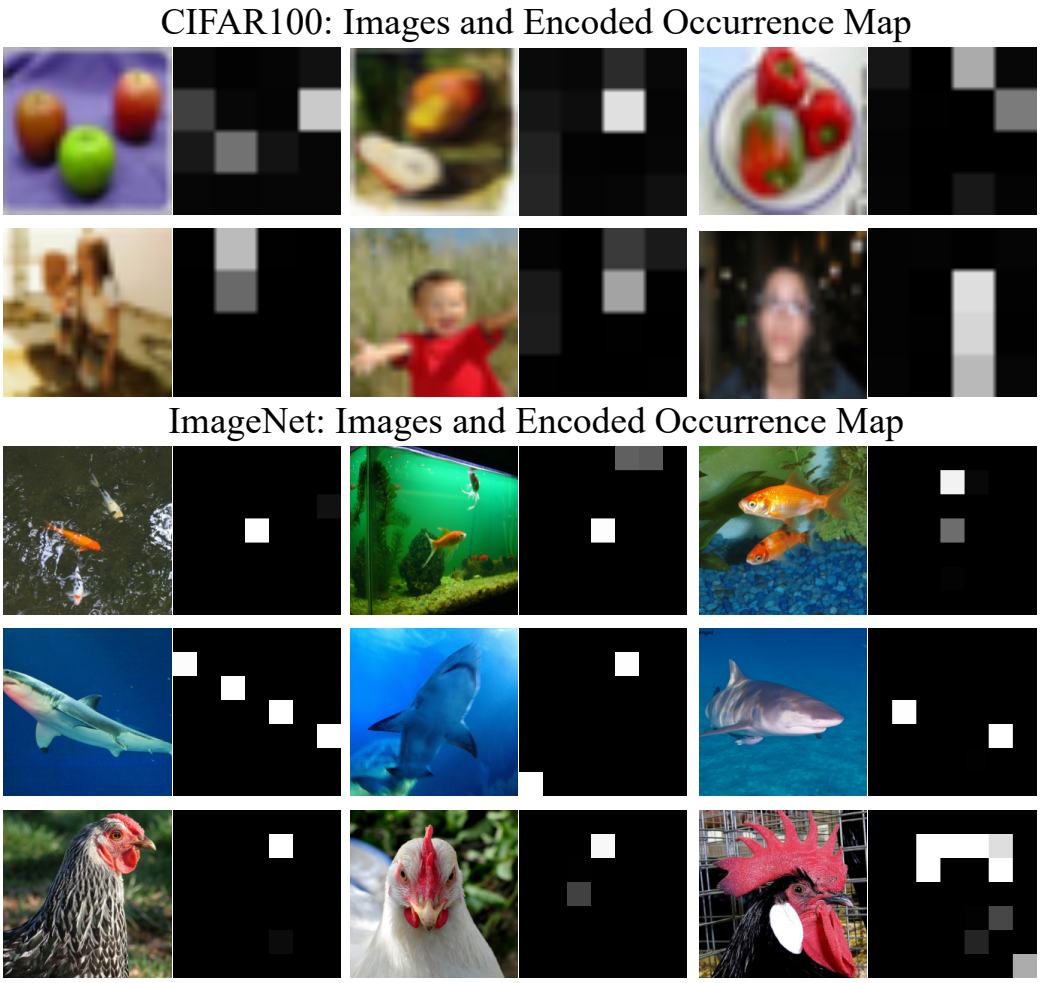


Figure 2.3: Encoded part occurrence maps discovered on CIFAR100 and ImageNet. Each row represents a different part.

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).

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 [He+16] 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 trained end-to-end on ImageNet using supervised training for comparison.

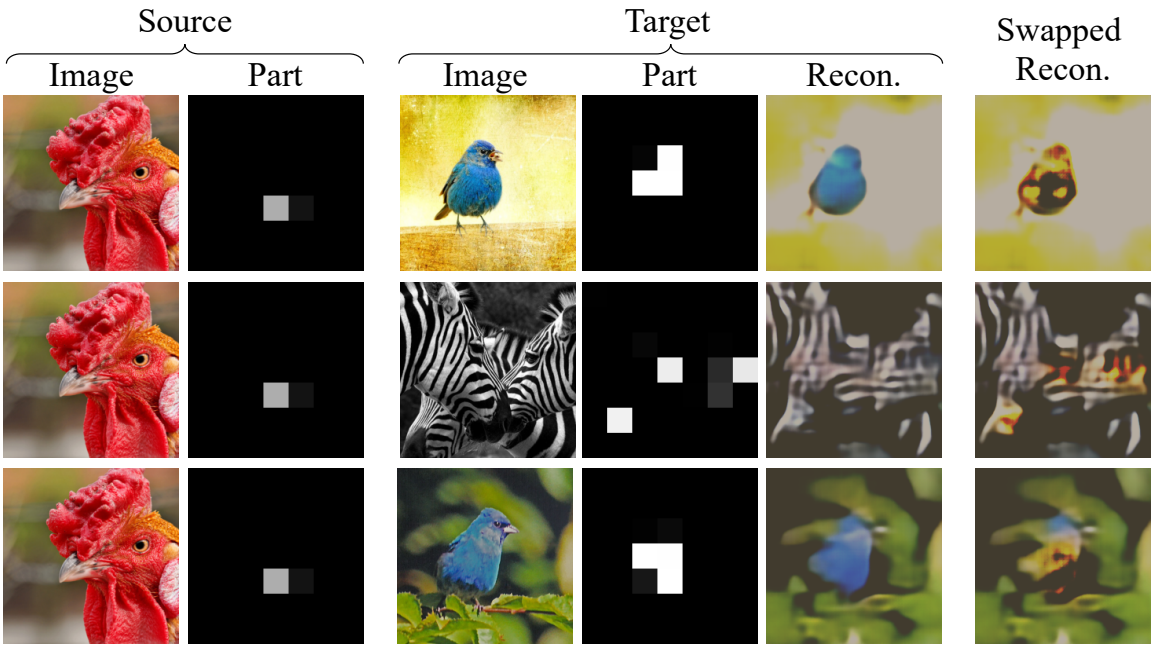


Figure 2.5: Swapping source and target part appearance. Column 1, 2 show a source image with the occurrence map of one of the parts. We can swap the appearance vector of this part with appearance vectors of a different part in target images. Column 3, 4 show three target images with occurrence maps of 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.

2.4.2 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.

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

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.

2.4.3 Ablation Studies

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 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 for Indoor67 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 $\mathbf{z}_{\text{occ}}^{\text{prior}}$. 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 β_{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}.

2.5 Conclusion

We presented a patch-based bottleneck in the VAE framework that encourages learning useful representations for recognition. Our method, PatchVAE, constrains the encoder architecture to only learn patches that are repetitive and consistent in images as opposed to learning *everything*, and therefore results in representations that perform much better for recognition tasks compared to vanilla VAEs. We also demonstrate that losses that favor high-energy foreground regions of an image are better for unsupervised learning of representations for recognition.

Chapter 3

PatchGame: Learning to Signal Mid-level Patches in Referential Games

3.1 Introduction

The ability to communicate using language is a signature characteristic of intelligence [Pin03]. 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 [MJB16].

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 [Cla96] 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. [Kri+18] argued the same viewpoint in the context of images. We, therefore, resort to the recent works in emergent language [Bat98; Kir02; Ste05; LPB16; HT17; Laz+18] 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 [Wit53]. 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 [Lew69]. 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. Fig. 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 [JT05], with a focus on low-level feature descriptors such as SIFT [Low04], SURF [BTV06], *etc.*. Recent works in deep learning have attempted to describe the entire

image with a fixed number of discrete symbols [OVK17; OLV18; ROV19], 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** [SGE12a; Jun+13; Doe+12b], the patches in the image that are both *representative* and *discriminative* [SGE12a; GSS20]. 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 [HT17; LPB16; Evt+17], we make the following key changes:

- Agents in the some of the prior works [HT17; LPB16; Evt+17] have access to a pre-trained network, such as AlexNet [KSH12] or VGG [SZ14], 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 [MH21].
- 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 [Den+09] images using a pre-trained Vision Transformer [Dos+20] 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 approaches [ROV19; GSS20] 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.

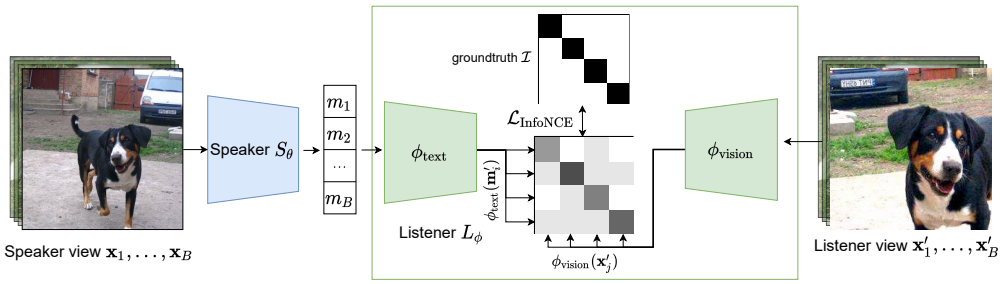


Figure 3.1: Overview of the Referential Game. We generate two random views of every image in the given batch of images. The speaker takes one of the views as the input and generates a sequence of symbols (or message). The listener takes the message sent by the speaker and the second view of the image,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

3.2 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 [Bat98; Kir02; Ste05; Wag+03; Bar+06; AKL17; AK17; Kaz+14]. 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 [Jor+16; Foe+16; Das+17; SSF16; Lee+17; Spi+17; LPB16; Laz+18] have employed deep learning methods to develop a discrete language for communication between the agents. Lazaridou, Peysakhovich, and Baroni [LPB16] 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. Evtimova et al. [Evt+17], Bouchacourt and Baroni [BB19], and Havrylov and Titov [HT17] relax this assumption and allow communication via variable-length sequences. Havrylov and Titov [HT17] allows the speaker agent to use an LSTM [HS97] to construct a variable-length message. Lazaridou et al. [Laz+18] and Havrylov and Titov [HT17] 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 [JGP16; MMT16] with the straight-through trick [BLC13] for learning a language in multi-agent environment was concurrently proposed by Mordatch and Abbeel [MA18] and Havrylov and Titov [HT17]. 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. [Low+19] and Keresztury and Bruni [KB20] 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 [And19] 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 [GSS20; OVK17; Rol16; Ram+21; ERO20; Nas+21; Gup+21; ROV19] without labels. The focus of VQ-VAE [OVK17] and VQ-VAE-2 [ROV19] 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 [OVK17; Sal+17b], or transformer [Vas+17; Rad+18] is learned on top of these symbols to sample new images. PatchVAE [GSS20] 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 [Car+21; Zbo+21; Che+20b; Car+20; CH20; MM20; He+20; Gri+20] 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 [MH21; MH19]. [MH19; MH21] extends Havrylov and Titov [HT17] by training a speaker and listener agents end-to-end without using pre-trained networks. However, the prior works [MH19; MH19; HT17] 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.

3.3 PatchGame

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.

3.3.1 Referential Game Setup

Fig. 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 [Lew69] played between two agents, a speaker S_θ and a listener L_ϕ as follows: As in the setting of Grill et al. [Gri+20], 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 [Soh16; OLV18].

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

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

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

where τ is a constant temperature hyperparameter.

The game setting used in our work is inspired from Lazaridou, Peysakhovich, and Baroni [LPB16] and Havrylov and Titov [HT17], 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

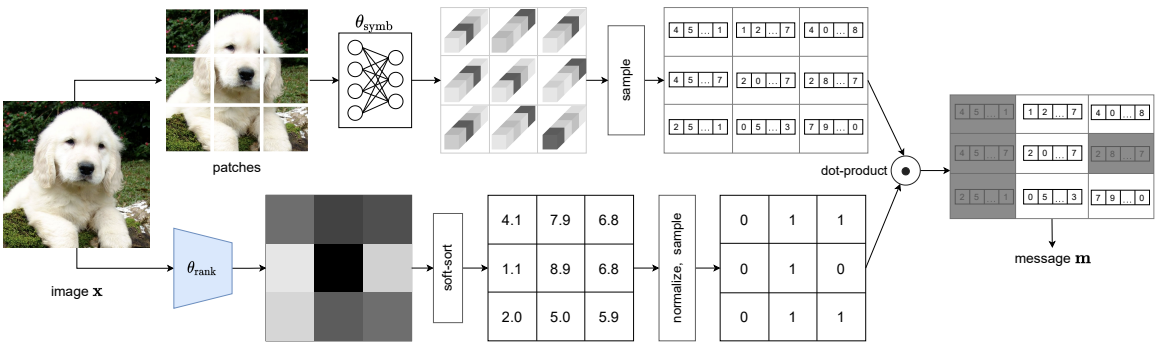


Figure 3.2: Speaker agent architecture. The upper branch represents the PatchSymbol, θ_{symb} module and the lower branch represents PatchRank, θ_{rank} module. Each of the modules take the raw image as the input. PatchSymbol computes a message for each patch of pre-defined size, using a fixed size vocabulary 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.

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.

3.3.2 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 [Evt+17; BB19; HT17] have achieved this by first converting the image into a continuous deterministic latent vector and then using an LSTM network [HS97] 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 [Laz+18; HT17], 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 [Doe+12b; Jun+13; SGE12a] 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 [JGP16; MMT16]. 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 [Gum54] and get a differentiable approximation of $\arg\max$ as follows:

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

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

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

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 [He+16] 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. [Blo+20] 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 [JGP16; MMT16] as

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

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

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 [Blo+20] for a detailed description of the soft-sort algorithm. Therefore, the final symbols encoded by the speaker agent, S_θ, is given by:

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

3.3.3 Listener agent architecture

As discussed in §3.3.1, 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 [Vas+17]. We prepend a $\langle\text{CLS}\rangle$ token at the beginning of each message sequence received from the speaker [Dev+18; Gup+21], and use the final embedding of $\langle\text{CLS}\rangle$ to compute the loss described in Eq. 3.3. We implement ϕ_{vision} using a small vision transformer [Dos+20], 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 [Zbo+21; Car+21; CH20], we add a high dimensional projection at the end of the last layer before computing the loss function.

3.3.4 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 [He+16] for θ_{rank}, a ResNet-18 [He+16] 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 [Den+09], 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 [BLC13]. We reiterate that the speaker and listener do not share weights, and the only supervision used is the InfoNCE loss defined in Eq. 3.3. Please refer to the appendix and code attached in the supplementary material for more details.

3.4 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. [Low+19], 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.

3.4.1 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 Fig. 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

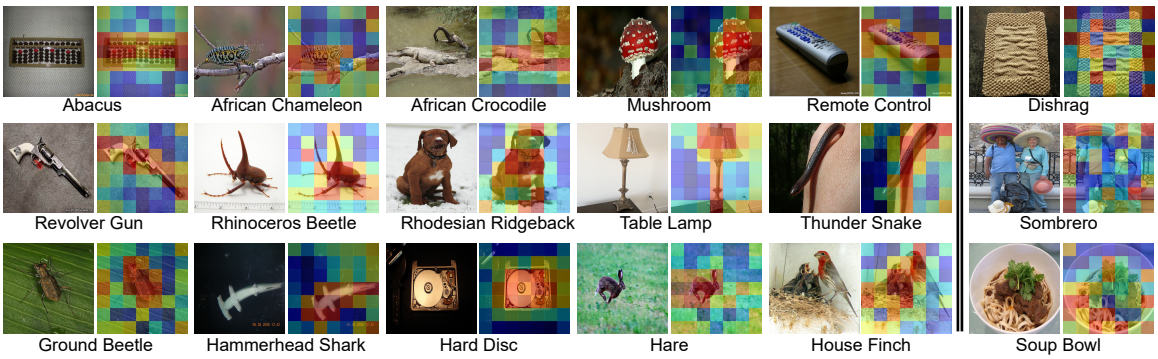


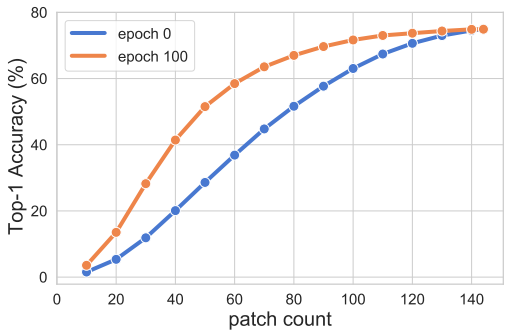
Figure 3.3: Heat maps based on Patch Importance. Red represents the most important patches and Blue the least important. Labels are listed for better understanding and are not used during training. We have used 224×224 images and 32×32 patches to get 49 non-overlapping patches per image. Model successfully 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.

and the background. Note that although approaches such as GradCam [Sel+17] 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 Fig. 3.3.

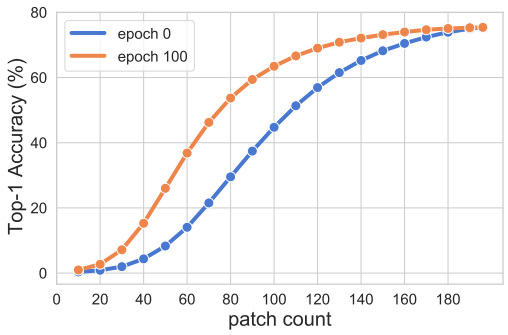
3.4.2 Positive Signaling - Image Classification with subset of patches provided by θ_{rank}

Recently proposed Vision Transformers (ViT) by Dosovitskiy et al. [Dos+20] 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 [Dos+20] 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 Fig. 3.4a and 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 [Dos+20]. 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.

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.

3.4.3 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 Fig. 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.

3.4.4 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

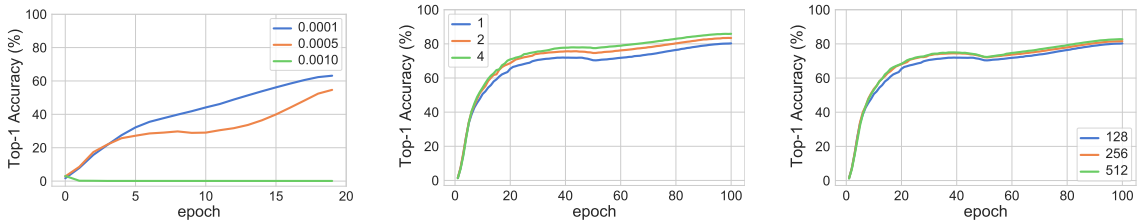
Table 3.1: Performance of Listener's Vision module ϕ_{vision} - Downstream classification accuracy for ImageNet dataset using k-NN (k=20)

Method	Top-1 (%)	Top-5 (%)
MoCo-v2 [He+20] (R18)	36.8 ($\pm$ 0.2)	60.3
VQ-VAE2 [OLV18]	17.2 ($\pm$ 0.6)	30.5
PatchVAE [GSS20] (R18, $S = 16$)	16.4 ($\pm$ 0.6)	28.5
PatchVAE [GSS20] (R18, $S = 32$)	21.3 ($\pm$ 0.5)	36.2
Ours (R18, $S = 16$)	27.6 ($\pm$ 0.6)	46.2
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 [He+20] (R18)	65.8
PatchVAE [GSS20] ($S = 16$)	52.2
PatchVAE [GSS20] ($S = 32$)	54.2
Ours ($S = 16$)	58.9
Ours ($S = 32$)	**61.3**

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 [He+20]. 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 approach continue to improve if we continue the training beyond 100 epochs (as also noted by He et al. [He+20]). Further, we use the listener's vision module as a pre-trained network for Pascal VOC dataset [Eve+15]. 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.

3.4.5 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 Fig. 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 Fig. 3.6b and 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 [Vas+17; Dev+18] is an $O(N^2)$ operation. A large vocabulary also increases the both the span of messages and computation cost linearly.

3.5 Discussion

Summary. In this work, we have shown that two cooperative neural network agents can develop a variable length communication protocol to solve a given task on a large scale image dataset with only self-supervision. To the best of our knowledge, this is the first work to develop an emergent communication via mid-level patches without using any pre-trained models or external labels. We have introduced a novel mid-level-patch based architecture for the speaker agent, in order to represent only the "interesting" parts of image with discrete symbols. The listener agent learns to align the messages and images using a language and image encoders to a common embedding space. The speaker and listener agents are trained end-to-end jointly to capture invariances induced by various data augmentations in order to solve a constrastive task. We propose a number of quantitative and qualitative measures to analyse and evaluate the emerged language. We also show two major applications of the developed approach - (1) extract representative and discriminative parts of the image (2) transfer learning in image classification tasks.

Limitations and Future Work. There are a few limitations of our approach. Firstly, one of the applications discussed in Section 3.4.1 is using fewer patches for inference in ViT. Although, using fewer patches reduces the memory cost of ViT, the overhead of using another neural network for predicting the important patches means our gain is minimal. In the future, we would like to explore even faster architectures to have a bigger impact on classification speed. Second, our method only works with fixed-size square patches. Discovering arbitrary sized mid-level patches is a challenging task that we would like to address in future work. Third, the language emerged from our current model is not grounded in natural language and requires human intervention for interpretation. Going forward, we would like to ground this emerged language in the natural language.

Broader Impact. Like most self-supervised approaches, our approach is data hungry and trained using a large amount of unlabeled data collected from the Internet. Our model in its current form is prone to learning and amplifying the biases present in the dataset, especially if the dataset in question is not carefully curated. While the data collection and labeling has been discussed in the original paper [Den+09], the community has focused towards imbalance and privacy violation in existing image datasets only recently [Yan+20]. A recent study [Yan+21] shows that 997 out of 1000 ImageNet categories are not 'people' categories. To compare against previous methods and allow future benchmarking, we

provide results on the 2012 version of the dataset with 1.28 million images in training set and 50000 images in the validation set.

Acknowledgements. We thank Matt Gwilliam, Max Ehrlich, and Pulkit Kumar for reviewing early drafts of the paper and helpful comments. This project was partially funded by DARPA SAIL-ON (W911NF2020009), and Amazon Research Award to AS.

Chapter 4

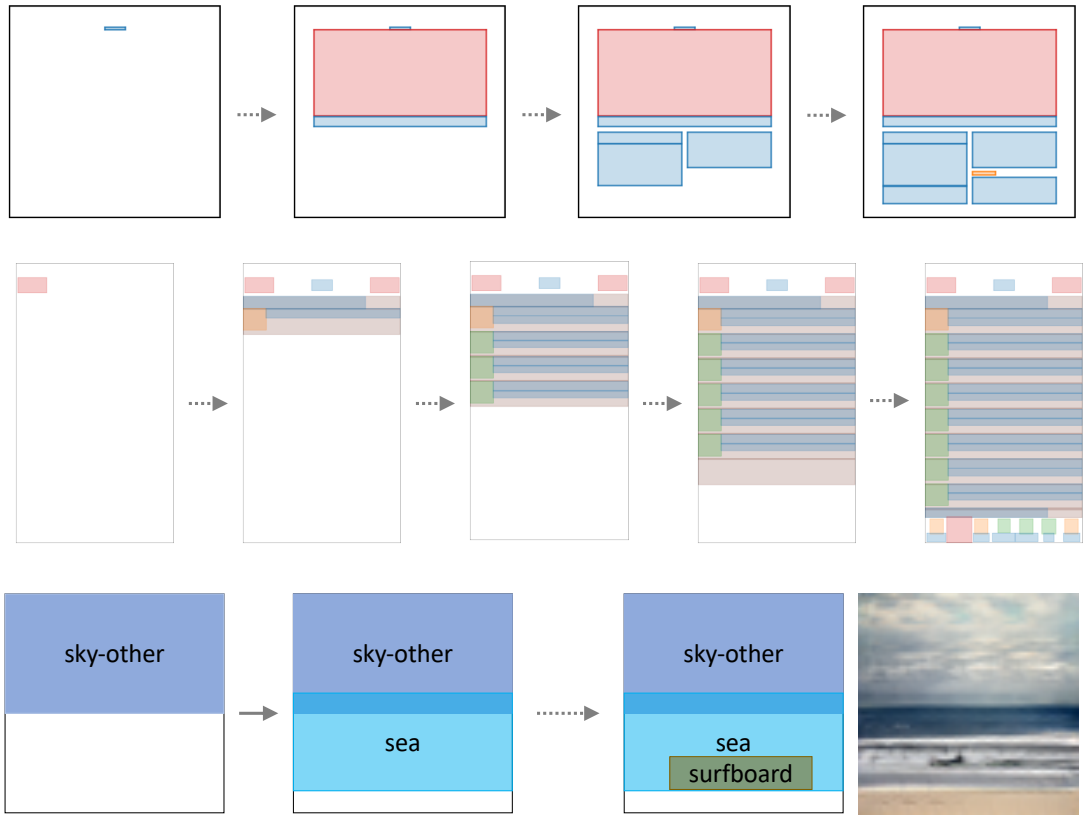
LayoutTransformer: Layout Generation and Completion with Self-attention

4.1 Introduction

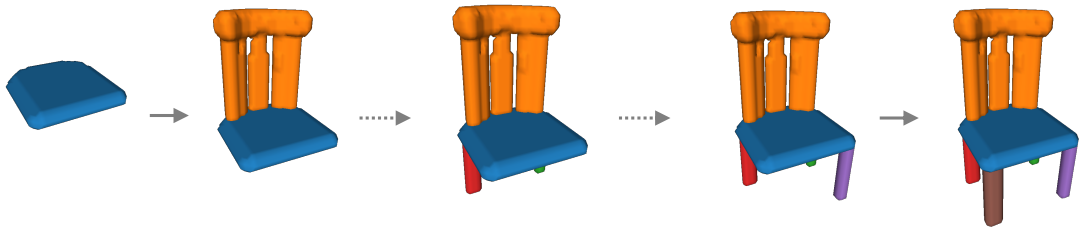
In the real world, there exists a strong relationship between different objects that are found in the same environment [TS01; SG16]. For example, a dining table usually has chairs around it, a surfboard is found near the sea, horses do not ride cars *etc*. [Bie81] provided strong evidence in cognitive 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 [KLA19; Kar+17], or animals [BDS18; Zha+18]. 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). 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 create layouts can potentially help generate synthetic data for various tasks tasks [Yan+17; CM17; Cha+15; WTK17; Wu+17]. Fig. 4.1 shows some of the layouts autoregressively generated by our approach in various domains such as documents, mobile



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



(b) Generating 3D objects autoregressively

Figure 4.1: 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.

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 [Mik+13]). 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 [Li+19b], Rico Mobile App Wireframes [Dek+17], PubLayNet Documents [ZTY19], and COCO Bounding Boxes [Lin+14]. 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.

4.2 Related Work

Generative models. Deep generative models based on CNNs such as variational auto-encoders (VAEs) [KW13a], and generative adversarial networks (GANs) [Goo+14] 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 [OK16; Che+20a] even when the data has no natural order [VBK15]. Many of these approaches often rely on low-level information [GSS20] such as pixels while generating images [BDS18; KLA19], videos [VPT16], or 3D objects [Wu+16; Yan+19; Par+19a; Gup+20] and not on semantic and geometric structure in the data.

Scene generation. Generating 2D or 3D scenes conditioned on sentence [Li+19d; Zha+17; Ree+16], a scene graph [JGF18; Li+19a; AW19], a layout [Don+17; HHW19; Iso+16;

Wan+18b] or an existing image [Lee+18] has drawn a great interest in vision community. Given the input, some works generate a fixed layout and diverse scenes [Zha+19], while other works generate diverse layouts and scenes [JGF18; Li+19d]. 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 [JGF18; Li+19d] that have layout generation as an intermediate step. We also demonstrate the results of our model with Layout2Im [Zha+19] for image generation.

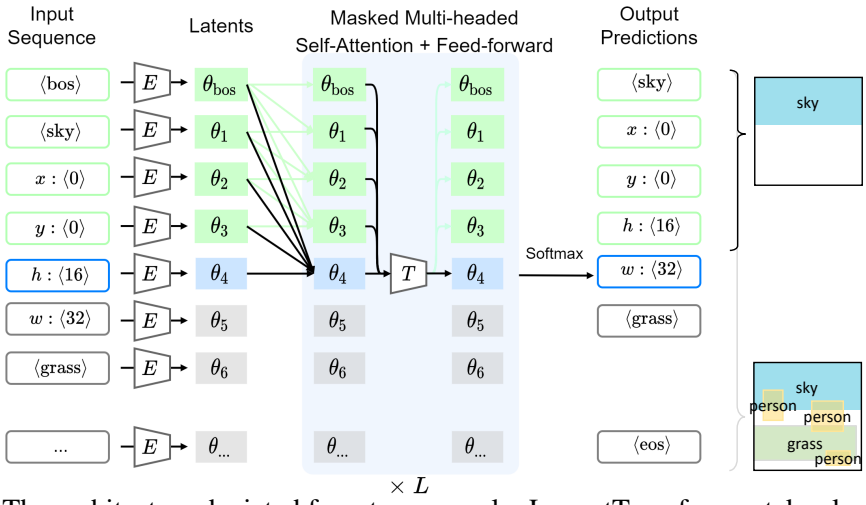


Figure 4.2: The architecture depicted for a toy example. LayoutTransformer takes layout elements as input and predicts the next layout elements as output. During training, we use teacher forcing, *i.e.*, use the ground-truth layout tokens as input to a multi-head decoder block. The first layer of this 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.

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, [Wan+18a; Wan+19; Li+19c; RWL19] 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 [Li+19b] uses a GAN framework to generate semantic and geometric properties of a fixed number of scene elements. LayoutVAE [Jyo+19] 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. [Zhe+19] attempt to generate document layouts given the images, keywords, and category of the document. [Pat+19] proposes a method to construct hierarchies of document layouts using a recursive variational autoencoder and sample new hierarchies to generate new document layouts. [MRC20] develops an auto-encoding framework for layouts using Graph Networks. 3D-PRNN [Zou+17], PQ-Net [Wu+20] and ComplementMe [Sun+17b], 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 §4.3.1) 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.

4.3 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 [Vas+17] 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 work with variable length layouts.

4.3.1 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 multidimensional 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++ [Sal+17c], 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:

$$p(\boldsymbol{\theta}_{1:5n+2}) = \prod_{j=1}^{5n+2} p(\boldsymbol{\theta}_j | \boldsymbol{\theta}_{1:j-1}) \tag{4.1}$$

Figure 4.3: **Generated 3D objects.** Top row shows input primitives to the model. Bottom row shows the layout obtained by our approach.

4.3.2 Model architecture and training

Our overall architecture is shown in Fig. 4.2. 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 [Vas+17] & 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 [He+16] and LayerNorm [BKH16].

$$\hat{\boldsymbol{\theta}}_j = \text{LayerNorm}(\boldsymbol{\theta}_j^{l-1} + \mathbf{h}^{\text{attn}}(\boldsymbol{\theta}_1^{l-1}, \ldots, \boldsymbol{\theta}_{j-1}^{l-1})) \tag{4.2}$$

$$\boldsymbol{\theta}_j^l = \text{LayerNorm}(\hat{\boldsymbol{\theta}}_j + \mathbf{h}^{\text{fc}}(\hat{\boldsymbol{\theta}}_j)) \tag{4.3}$$

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 [Sze+16], 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 [Yu+19] 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 [CZ19]'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 [VBK15]. 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 [Jyo+19; Li+19d; Hon+18; Wan+18a], image generation [Sal+17c; Gre+15] and 3D shape generation [Wu+20; Zou+17]. 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 [KB14] 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 [Hol+19], with top-$p = 0.9$ which has been shown to perform better as compared to greedy sampling and beam search [STN94].

4.4 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.

4.4.1 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 [Wu+20]. 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. 4.3 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.

Table 4.1: 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 [Yan+19]	1.74	2.42	<u>7.87</u>	46.83	46.98	<u>60.88</u>	<u>59.89</u>
StructureNet [Mo+19]	4.77	0.97	15.24	29.67	31.7	75.32	74.22
IM-Net [CZ19]	0.84	**0.74**	12.28	52.35	54.12	68.52	67.12
PQ-Net [Wu+20]	<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**

Quantitative Evaluation. The output of our model is point clouds sampled on the surface of 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 4.1 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.

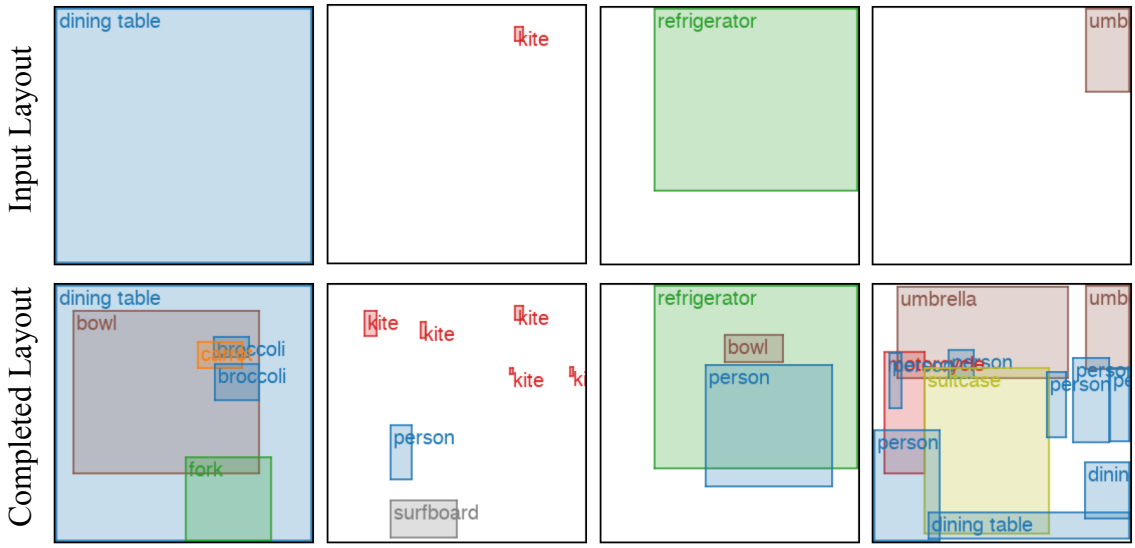


Figure 4.4: **Generated layouts.** Top row shows seed layouts input to the model. Bottom row shows the layout obtained with nucleus sampling. We skip the 'stuff' bounding boxes for clarity.

4.4.2 Layouts for natural scenes

COCO bounding boxes dataset is obtained using bounding box annotations in COCO Panoptic 2017 dataset [Lin+14]. We ignore the images where the *isCrowd* flag is true following the LayoutVAE [Jyo+19] 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 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[Li+19b], LayoutVAE [Jyo+19], ObjGAN [Li+19d], and sg2im [JGF18]. 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 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. 4.4 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 [Zha+19].

Semantics Emerge via Layout. We posited earlier that capturing layout should capture contextual relationships between various elements. We provide further evidence of our

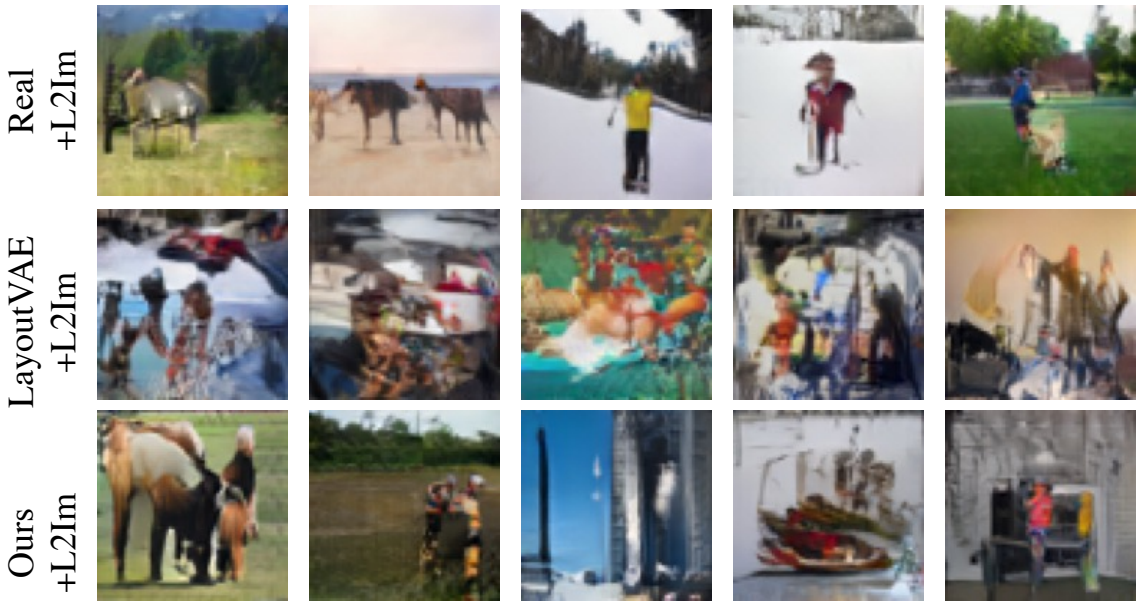


Figure 4.5: **Downstream task.** Image generation with layouts [Zha+19]. FID and IS scores for the generated images provided in Table 4.3.

argument in Fig. 4.6. 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 [GSH19] who use visual co-occurence to learn category embeddings. Table 4.2 shows word2vec [Mik+13] 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. 4.7. $y-$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 [Zha+19] 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

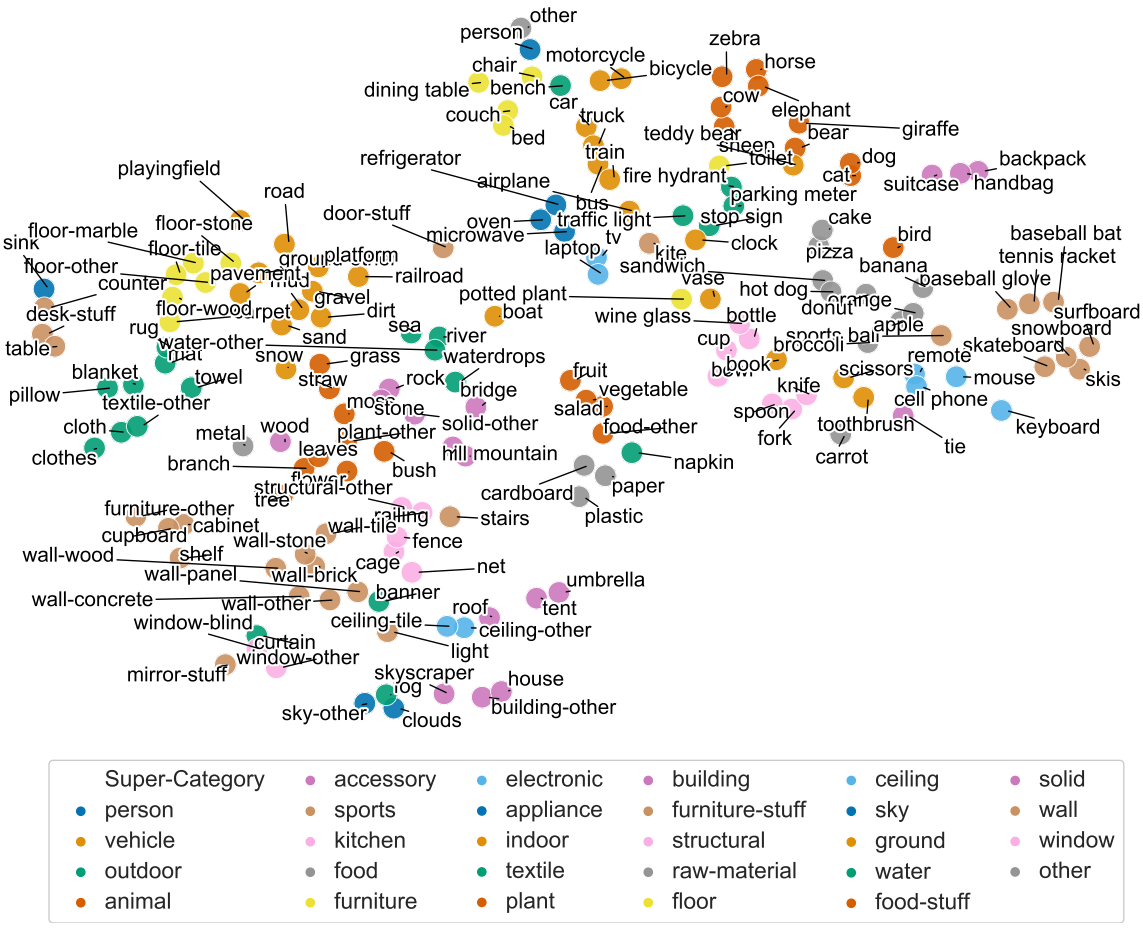


Figure 4.6: TSNE plot of learned category embeddings. Words are colored by their super-categories provided in the COCO. Observe that semantically similar categories cluster together. Cats and dogs are closer as compared to sheep, zebra, or cow.

LayoutGAN, which are unconditional. We clearly outperform LayoutGAN on IS and FID metrics.

4.4.3 Layouts for Apps and Documents

Rico Mobile App Wireframes. Rico mobile app dataset [Dek+17; Liu+18] 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 selected layouts for validation and 15% for testing.

PubLayNet. PubLayNet [ZTY19] 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

Table 4.2: **Analogies**. We demonstrate linguistic nuances being captured by our category embeddings by attempting word2vec [Mik+13] 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

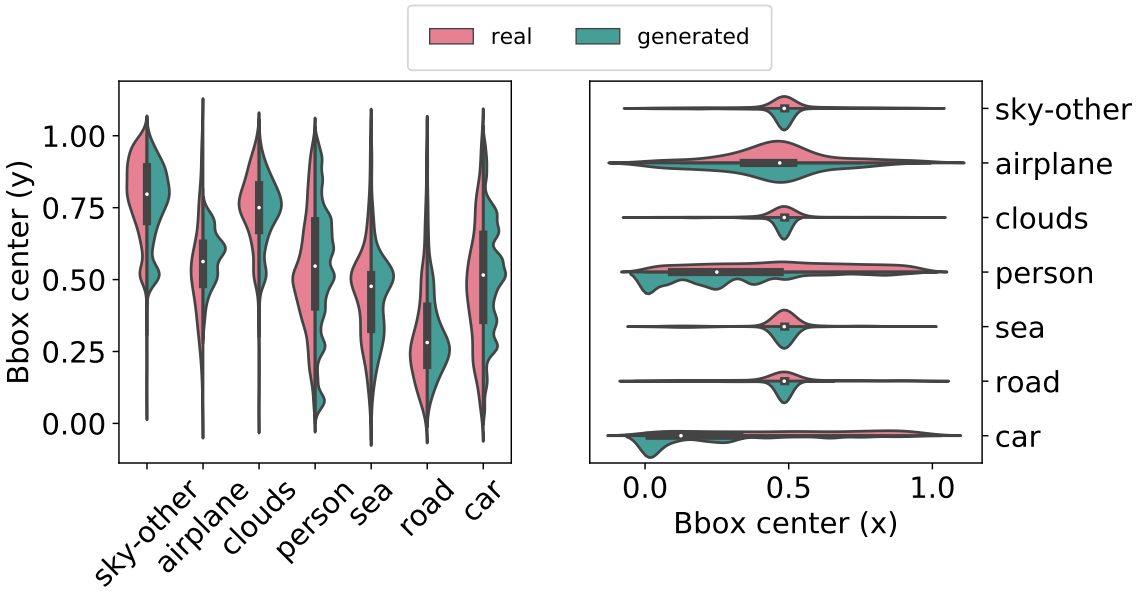


Figure 4.7: Distribution of xy-coordinates of bounding boxes centers. Distributions for generated and real layouts is similar. The y-coordinate tends to be more informative (*e.g.*, sky on the top, road and sea at the bottom)

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. 4.8 and 4.10 shows some of the generated samples of our model 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. 4.9 shows multiple completions done by our model for the same initial element.

Comparison with baselines. We use the same baselines discussed in §4.4.2. Fig. 4.10 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

Table 4.3: **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 [Jyo+19] to compute. We also generated images from layout using [Zha+19] and compute IS and FID. Following [JGF18], 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 [Li+19b]	-	3.2 (0.22)	89.6 (1.6)
LayoutVAE [Jyo+19]	3.29	7.1 (0.41)	64.1 (3.8)
ObjGAN [Li+19d]	5.24	7.5 (0.44)	62.3 (4.6)
sg2im [JGF18]	3.4	3.3 (0.15)	85.8 (1.6)
Ours	**2.28**	**7.6 (0.30)**	**57.0 (3.5)**

Figure 4.8: **RICO layouts.** Generated layouts for the RICO dataset. We skip the categories of bounding boxes for the sake of clarity.

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 4.4 shows that layouts generated by our method resemble real data statistics better than LayoutGAN and LayoutVAE.

4.4.4 Failure Cases

Our model has a few failure cases, *e.g.*, in Fig. 4.3 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. 4.7, which shows difference in distribution of real & generated layouts for persons and cars.

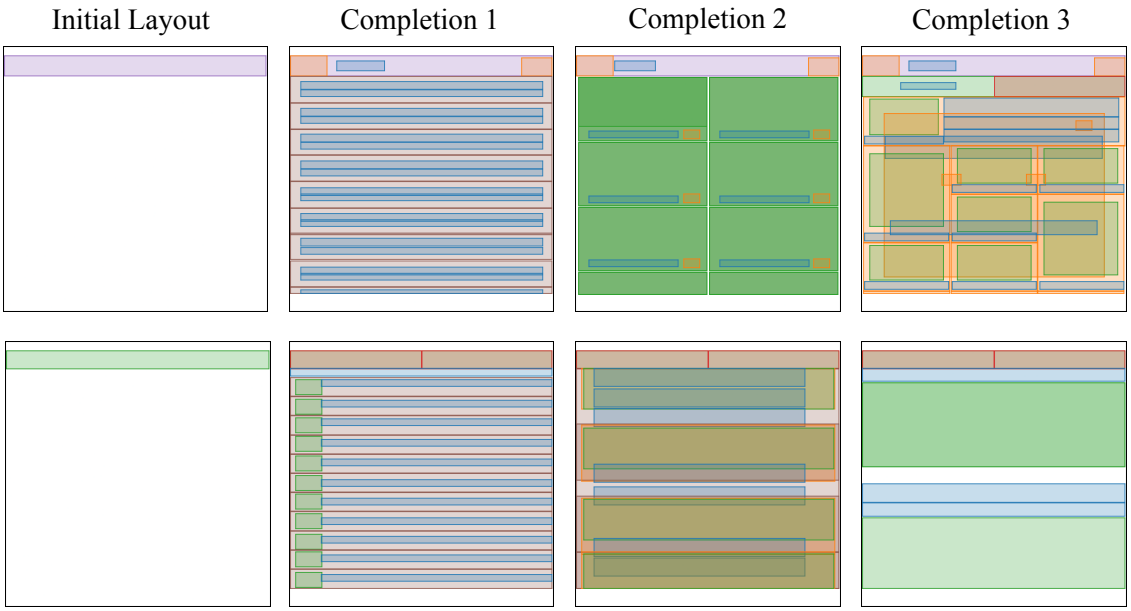


Figure 4.9: Multiple completions from same initial element

Table 4.4: 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 [JGF18]	7.43	25.2 (46)	16.5 (31)	7.12	30.2 (26)	3.4 (12)
ObjGAN [Li+19d]	4.21	39.2 (33)	36.4 (29)	4.20	38.9 (12)	8.2 (7)
LayoutVAE [Jyo+19]	2.54	41.5 (29)	34.1 (27)	2.45	40.1 (11)	14.5 (11)
LayoutGAN [Li+19b]	-	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)

4.5 Conclusion.

We propose LayoutTransformer, a self-attention framework to generate layouts of graphical elements. Our model uses self-attention model to capture contextual relationship between different layout elements and generate novel layouts, or complete partial layouts. By modeling layout primitives as joint distribution of composable attributes, our model performs competitively with the state-of-the-art approaches for very diverse datasets such as Rico Mobile App Wireframes, COCO bounding boxes, PubLayNet documents, and 3D shapes. We perform a comprehensive qualitative and quantitative evaluation of our model in various domains. We will release our code and models and hope that our model will provide a good starting point for layout modeling applications in various data domains.

There are a few noteworthy limitations of our approach. First, our model requires a layout or a scene to be decomposed into compositional primitives. In many cases, such

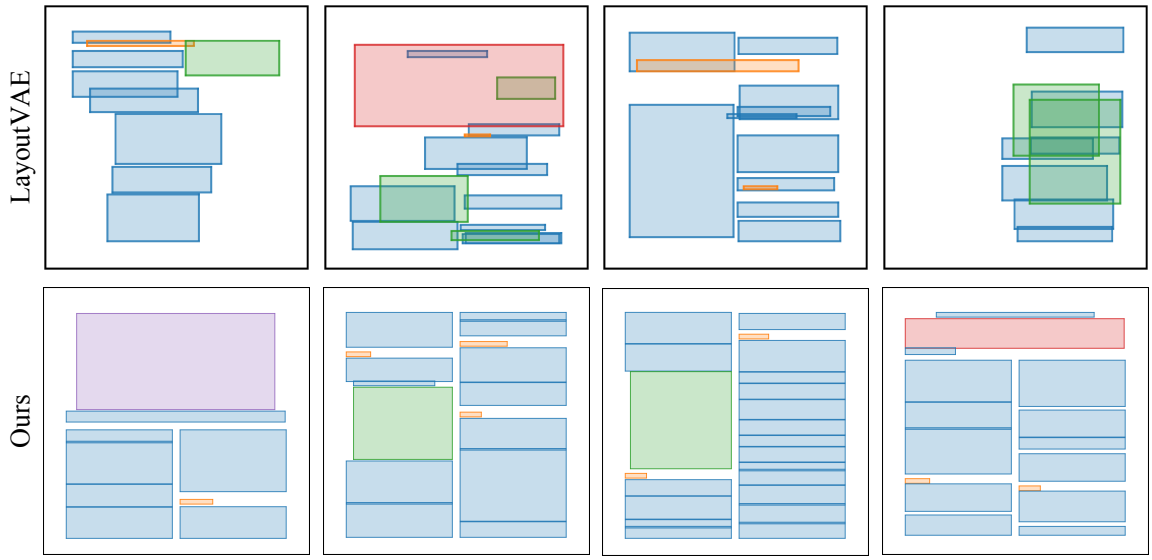


Figure 4.10: **Document Layouts.** Generated samples LayoutVAE (top) and our method (bottom). Our method produces aligned bounding boxes for various elements.

primitives might not be even defined. Second, like most data-driven approaches, generated layouts are dominated by high frequency objects or shapes in the dataset. We can control the diversity to some extent using improved sampling techniques, however, generating diverse layouts that not only learn from data, but also from human priors or pre-defined rules is an important direction of research which we will continue to explore.

Acknowledgements. We thank Luis Goncalves, Yuting Zhang, Gowthami Somepalli, and Pulkit Kumar for helpful discussions and reviewing early drafts of the paper. This project was partially funded by the DARPA SAIL-ON (W911NF2020009) program.

Chapter 5

Proposed Work

5.1 Introduction

Real world applications of 3D content creation such as designing a game level or a video often involve teams of professionals with years of expertise, a large database of digital 3D assets, and a long development cycle. In the workflow of a game level design for instance, 3D designer(s) often make a iteratively manipulate the scene geometry making numerous local adjustments to the scene after which another team of professionals add texture and fines geometric details to the initial design provided by the designer. This work proposes a method to add textures to geometric shapes represented by 3D meshes.

In recent years, there has been a lot of advances in 2D image synthesis, as well as image-to-image translation techniques [Kar+17; KLA19; Par+19b; Iso+16; OK16; Sal+17c; Sal+17b]. Even though surfaces of most real-world objects are 2D manifolds, the advances on 2D euclidean grids, *i.e.*, the images, hasn't translated to the 3D surfaces. Note that although the recently proposed neural fields [Mil+20] have arguably shown best visual quality when synthesizing novel views, the intermediate 3D representation used by these class of techniques is implicit and cannot be easily controlled by artists. In this work, we restrict our focus to explicit representations of 3D scene geometry that allow artists to edit different elements of the scene freely. We propose a formulation to synthesize textures on the surfaces of objects. The proposed formulation works on any polygonal meshes and doesn't assume any requirements on genus, or watertightness of the mesh.

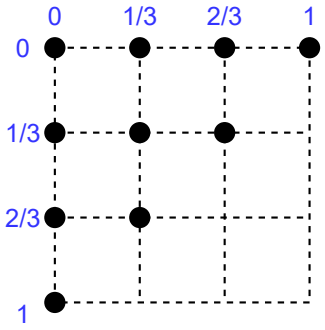
In the next section, we discuss the various steps of the proposed framework, and some preliminary results.

5.2 Approach

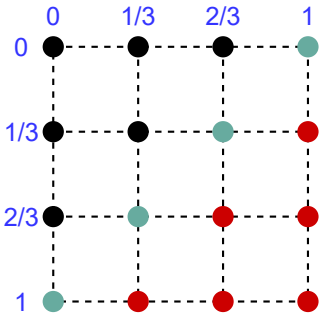
The key step in our approach is to disentangle the texture representation from the texture synthesis. One of the most common ways to represent the texture for 3D models is to have texture map (which is usually created by professional artist) and assign texture coordinates to vertices of a mesh. Instead we propose to represent the texture directly in the 3D model space using Yuksel, Keyser, and House [YKH10]. We further quantize and compress the Mesh Colors using VQ approach proposed by Oord, Vinyals, and Kavukcuoglu [OVK17].

Finally, we propose to learn an auto-regressive model conditioned on the mesh geometry.

5.2.1 Mesh Colors



(a) 4×4 Mesh Colors for a face



(b) 4×4 Mesh Colors for an edge

Figure 5.1: Mesh Color [YKH10] vertices on (a) a triangle and (b) a quadrilateral. A quadrilateral can be viewed as a pair of two triangles sharing an edge.

Fig. 5.1 shows the positions within a triangle or a quad where we sample the Mesh Colors. Since the mesh colors are sampled directly on the 3D model space, there are no seams and no need of defining texture coordinates.

(a) Pair of faces

(b) Face Texture

(c) Edge Texture

Figure 5.2: Fig. 5.2a shows a pair of triangles on a mesh. We can either store a single triangle's texture per image as show in Fig. 5.2b which is simple but wasteful representation, or store a pair of triangles per image as shown in Fig. 5.2c. Working with a pair of triangles is a memory efficient but more challenging since features need to be computed at the edge level and the two triangles need not be on the same plane.

While we can store mesh colors for different triangles at different resolutions, for simplicity we assume a fixed resolution for all the triangles. We further pack each triangle or a quad into an image. Fig. 5.2 shows an example of how we pack texture for two triangles on a cottage roof.

5.2.2 Vector Quantized Mesh Colors

Using a fixed resolution for all triangles can be memory intensive since a mesh can easily have hundreds of thousands of triangles. Hence we further propose to compress the texture

using vector quantization as shown in Figure 5.3. Starting with a $W \times W$ image, a CNN Encoder downsizes the image by a factor of 16 so that image represented by $\frac{W^2}{16}$ vectors. These vectors are shared across multiple images by defining a codebook of fixes size. For each of the $\frac{W^2}{16}$ vectors representing the image, we replace the vector with a closest vector from the codebook and pass it to the decoder. The decoder decodes the image using these new vectors instead from the codebook. The codebook is learned during training.

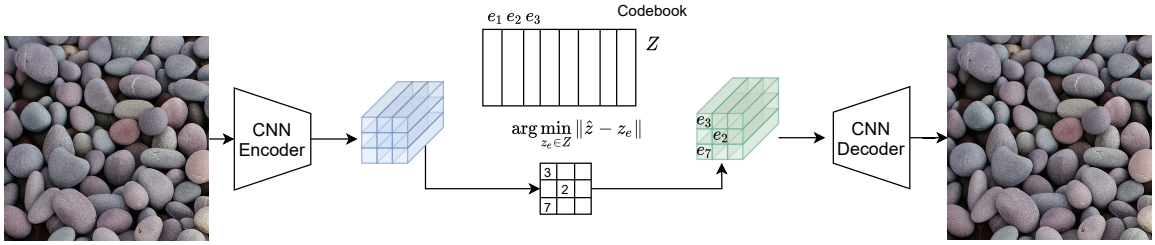


Figure 5.3: **Vector Quantization for textures**. Starting with a $W \times W$ image, a CNN Encoder downsizes the image by a factor of 16 so that image represented by $\frac{W^2}{16}$ vectors. These vectors are shared across multiple images by defining a codebook of fixes size. For each of the $\frac{W^2}{16}$ vectors representing the image, we replace the vector with a closest vector from the codebook and pass it to the decoder. The decoder decodes the image using these new vectors instead from the codebook. The codebook is learned during training.

5.2.3 Triangles to Textures

Face Spirals. In order to sample the colors (or texture) of a triangle, we consider the triangles in the neighborhood. To obtain this neighborhood, we consider a spiral of triangles around the given triangle. Figure 5.4 shows an example of a patch spiral obtained around a triangle.

Geometric Features. For each triangle, we compute the geometric features in the form of fourier feature embeddings of the vertex coordinates and vertex normals of a face. These geometric features would be used in the next step as a input for texture synthesis.

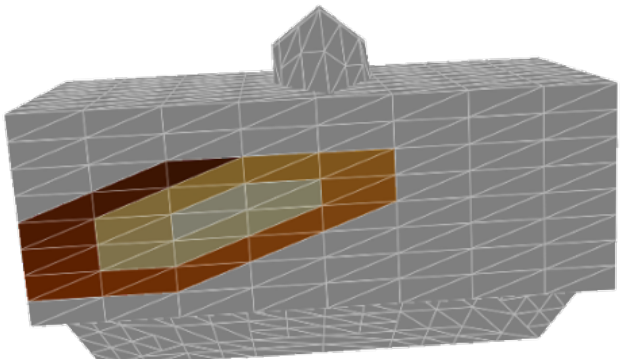


Figure 5.4: We use a variation of Lim et al. [Lim+18] to sample a localized patch of triangles on the mesh.

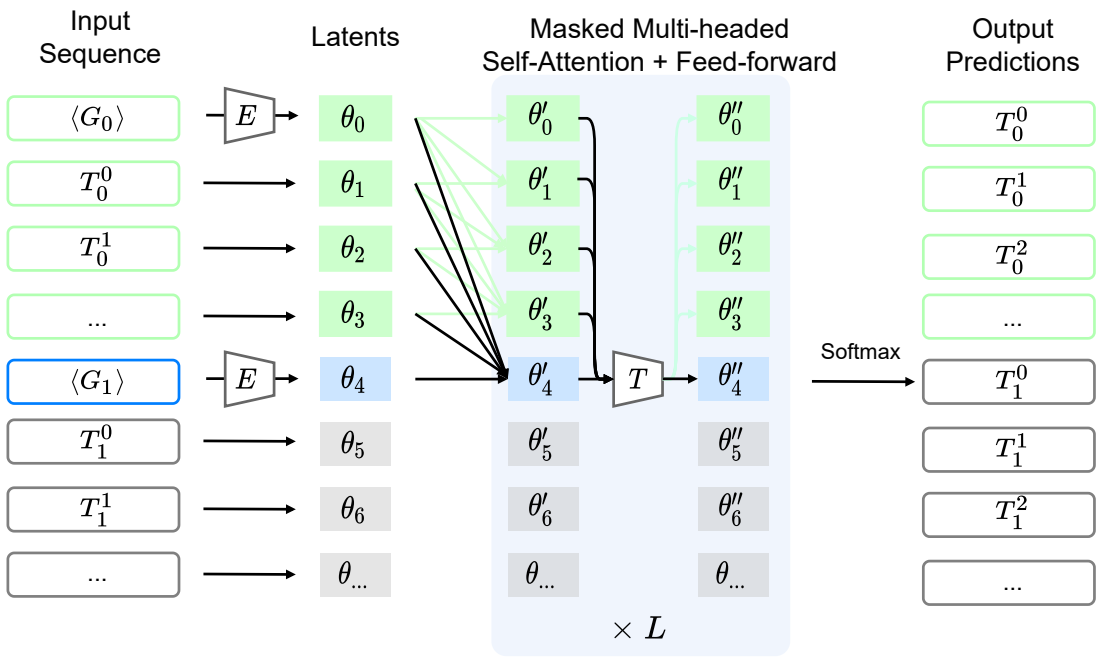


Figure 5.5: Given a patch of triangles, we represent each triangle using geometric features G and quantized textures T. In order to make prediction for a texture token, a GPT looks at the embeddings of previous triangles including the geometry and textures.

Texture Transformer. We next propose to use a texture transformer conditioned on the mesh geometry in order to learn the distribution of quantized textures on mesh surface. Figure 5.5 shows the architecture diagram. Figure 5.6 shows the output of model (right) when trained on a single input mesh texture (left).

(a) Source textured mesh

(b) Learned textured overlaid on the geometry

Figure 5.6: **Left**: Input textured mesh. **Right**: Textured predicted by the GPT trained on the same mesh.

Appendix A

Reading List

A.1 Image Representation

1. Lowe, David G. Object recognition from local scale-invariant features. ICCV, 1999.
2. Singh, Saurabh, et al. Unsupervised discovery of mid-level discriminative patches. ECCV, 2012.
3. Juneja, Mayank, et al. Blocks that shout: Distinctive parts for scene classification. CVPR, 2013.
4. Noroozi, Mehdi, et al. Unsupervised learning of visual representations by solving jigsaw puzzles. ECCV, 2016.
5. Donahue, Jeff, et al. Adversarial feature learning. ICLR, 2017.
6. Lazaridou, Angeliki, et al. Multi-agent cooperation and the emergence of (natural) language. ICLR, 2017.
7. Havrylov, Serhii, et al. Emergence of language with multi-agent games: Learning to communicate with sequences of symbols. NeurIPS 2017.
8. Chen, Ting, et al. A simple framework for contrastive learning of visual representations. ICML, 2020.
9. He, Kaiming, et al. Momentum contrast for unsupervised visual representation learning. CVPR, 2020.

A.2 Image and Texture Synthesis

1. Heeger, David J. et al. Pyramid-based texture analysis/synthesis. TOG. 1995.
2. Efros, Alexei A., et al. Image quilting for texture synthesis and transfer. SIGGRAPH, 2001.

3. Barnes, Connelly, et al. PatchMatch: A randomized correspondence algorithm for structural image editing. TOG, 2010.

4. Kingma, Diederik P., et al. Auto-encoding variational bayes. ICLR, 2014.

5. Goodfellow, Ian, et al. Generative adversarial networks. NeurIPS 2014.

6. Gatys, Leon, et al. Texture synthesis using convolutional neural networks. NeurIPS, 2015.

7. Oord, Aaron van den, et al. Neural discrete representation learning. NeurIPS, 2017.

8. Isola, Phillip, et al. Image-to-image translation with conditional adversarial networks. CVPR, 2017.

9. Karras, Tero, et al. A style-based generator architecture for generative adversarial networks. CVPR, 2019.

10. Esser, Patrick, et al. Taming transformers for high-resolution image synthesis. CVPR, 2021.

11. Liu, Andrew, et al. Infinite nature: Perpetual view generation of natural scenes from a single image. ICCV, 2021.

A.3 Composition of Primitives

1. Kalogerakis, Evangelos, et al. A probabilistic model for component-based shape synthesis. TOG, 2012.

2. Doersch, Carl, et al. What makes paris look like paris?. TOG, 2012.

3. Tulsiani, Shubham, et al. Learning shape abstractions by assembling volumetric primitives. CVPR. 2017.

A.4 Geometric Deep Learning

1. Wu, Jiajun, et al. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. NeurIPS, 2016.

2. Park, Jeong Joon, et al. DeepSDF: Learning continuous signed distance functions for shape representation. CVPR, 2019.

3. Nash, Charlie, et al. Polygen: An autoregressive generative model of 3d meshes. ICML, 2020.

4. Hertz, Amir, et al. Deep geometric texture synthesis. SIGGRAPH, 2020.

5. Mildenhall, Ben, et al. NeRF: Representing scenes as neural radiance fields for view synthesis. ECCV, 2020.

6. Hao, Zekun, et al. "GANcraft: Unsupervised 3D Neural Rendering of Minecraft Worlds." ICCV, 2021.

7. Bronstein, Michael M., et al. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. Book (2021).

Bibliography

[AGW18] Lama Affara, Bernard Ghanem, and Peter Wonka. "Supervised Convolutional Sparse Coding". In: *CoRR* abs/1804.02678 (2018). arXiv: 1804.02678. URL: http://arxiv.org/abs/1804.02678.

[Agu+17] Eirikur Agustsson et al. "Soft-to-hard vector quantization for end-to-end learning compressible representations". In: *Advances in Neural Information Processing Systems*. 2017, pp. 1141–1151.

[And19] Jacob Andreas. "Measuring compositionality in representation learning". In: *arXiv preprint arXiv:1902.07181* (2019).

[AK17] Jacob Andreas and Dan Klein. "Analogs of linguistic structure in deep representations". In: *arXiv preprint arXiv:1707.08139* (2017).

[AKL17] Jacob Andreas, Dan Klein, and Sergey Levine. "Learning with latent language". In: *arXiv preprint arXiv:1711.00482* (2017).

[ACB17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein gan". In: *arXiv preprint arXiv:1701.07875* (2017).

[AW19] Oron Ashual and Lior Wolf. "Specifying object attributes and relations in interactive scene generation". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 4561–4569.

[BKH16] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. "Layer normalization". In: *arXiv preprint arXiv:1607.06450* (2016).

[Bar+06] Andrea Baronchelli et al. "Sharp transition towards shared vocabularies in multi-agent systems". In: *Journal of Statistical Mechanics: Theory and Experiment* 2006.06 (2006), P06014.

[Bat98] John Batali. "Computational simulations of the emergence of grammar". In: *Approach to the Evolution of Language* (1998), pp. 405–426.

[BTV06] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. "Surf: Speeded up robust features". In: *European conference on computer vision*. Springer. 2006, pp. 404–417.

[BLC13] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. "Estimating or propagating gradients through stochastic neurons for conditional computation". In: *arXiv preprint arXiv:1308.3432* (2013).

[Bie81] Irving Biederman. "On the semantics of a glance at a scene". In: *Perceptual organization* 213 (1981), p. 253.

[Blo+20] Mathieu Blondel et al. “Fast differentiable sorting and ranking”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 950–959.

[BB19] Diane Bouchacourt and Marco Baroni. “Miss Tools and Mr Fruit: Emergent communication in agents learning about object affordances”. In: *arXiv preprint arXiv:1905.11871* (2019).

[BDS18] Andrew Brock, Jeff Donahue, and Karen Simonyan. “Large scale gan training for high fidelity natural image synthesis”. In: *arXiv preprint arXiv:1809.11096* (2018).

[CM17] Samuele Capobianco and Simone Marinai. “DocEmul: a Toolkit to Generate Structured Historical Documents”. In: *CoRR* abs/1710.03474 (2017). arXiv: 1710.03474. URL: http://arxiv.org/abs/1710.03474.

[Car+20] Mathilde Caron et al. “Unsupervised learning of visual features by contrasting cluster assignments”. In: *arXiv preprint arXiv:2006.09882* (2020).

[Car+21] Mathilde Caron et al. “Emerging Properties in Self-Supervised Vision Transformers”. In: *arXiv preprint arXiv:2104.14294* (2021).

[Cha+15] Angel X. Chang et al. “Text to 3D Scene Generation with Rich Lexical Grounding”. In: *CoRR* abs/1505.06289 (2015). arXiv: 1505.06289. URL: http://arxiv.org/abs/1505.06289.

[Che+20a] Mark Chen et al. “Generative Pretraining from Pixels”. In: *Proceedings of the 37th International Conference on Machine Learning*. 2020.

[Che+20b] Ting Chen et al. “A simple framework for contrastive learning of visual representations”. In: *International conference on machine learning*. PMLR. 2020, pp. 1597–1607.

[CH20] Xinlei Chen and Kaiming He. “Exploring Simple Siamese Representation Learning”. In: *arXiv preprint arXiv:2011.10566* (2020).

[CZ19] Zhiqin Chen and Hao Zhang. “Learning implicit fields for generative shape modeling”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5939–5948.

[Cla96] Herbert H Clark. *Using language*. Cambridge university press, 1996.

[Das+17] Abhishek Das et al. “Learning cooperative visual dialog agents with deep reinforcement learning”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2951–2960.

[Dek+17] Biplab Deka et al. “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.

[Den+09] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *Computer Vision and Pattern Recognition, 2009. IEEE Conference on*. Ieee. 2009, pp. 248–255.

[Dev+18] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).

[DGE14] Carl Doersch, Abhinav Gupta, and Alexei A Efros. “Context as supervisory signal: Discovering objects with predictable context”. In: *European Conference on Computer Vision*. Springer. 2014, pp. 362–377.

[DGE15] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. “Unsupervised Visual Representation Learning by Context Prediction”. In: *International Conference on Computer Vision (ICCV)*. 2015.

[Doe+12a] Carl Doersch et al. “What Makes Paris Look like Paris?” In: *ACM Transactions on Graphics (SIGGRAPH)* 31.4 (2012), 101:1–101:9.

[Doe+12b] Carl Doersch et al. “What makes paris look like paris?” In: *ACM Transactions on Graphics* 31.4 (2012).

[DKD16] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. “Adversarial feature learning”. In: *arXiv preprint arXiv:1605.09782* (2016).

[Don+17] Hao Dong et al. “Semantic image synthesis via adversarial learning”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 5706–5714.

[Dos+20] Alexey Dosovitskiy et al. “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *arXiv preprint arXiv:2010.11929* (2020).

[ERO20] Patrick Esser, Robin Rombach, and Björn Ommer. “Taming Transformers for High-Resolution Image Synthesis”. In: *arXiv preprint arXiv:2012.09841* (2020).

[Eve+15] M. Everingham et al. “The Pascal Visual Object Classes Challenge: A Retrospective”. In: *International Journal of Computer Vision* 111.1 (Jan. 2015), pp. 98–136.

[Evt+17] Katrina Evtimova et al. “Emergent communication in a multi-modal, multi-step referential game”. In: *arXiv preprint arXiv:1705.10369* (2017).

[Foe+16] Jakob N Foerster et al. “Learning to communicate with deep multi-agent reinforcement learning”. In: *arXiv preprint arXiv:1605.06676* (2016).

[Fra+17] Marco Fraccaro et al. “A Disentangled Recognition and Nonlinear Dynamics Model for Unsupervised Learning”. In: *NIPS*. 2017.

[Goo+14] Ian Goodfellow et al. “Generative adversarial nets”. In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[Gre+15] Karol Gregor et al. “Draw: A recurrent neural network for image generation”. In: *arXiv preprint arXiv:1502.04623* (2015).

[Gri+20] Jean-Bastien Grill et al. “Bootstrap your own latent: A new approach to self-supervised learning”. In: *arXiv preprint arXiv:2006.07733* (2020).

[Gul+16] Ishaan Gulrajani et al. “Pixelvae: A latent variable model for natural images”. In: *arXiv preprint arXiv:1611.05013* (2016).

[Gum54] Emil Julius Gumbel. *Statistical theory of extreme values and some practical applications: a series of lectures*. Vol. 33. US Government Printing Office, 1954.

[GSS20] 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*. 2020, pp. 4746–4755.

[Gup+20] Kamal Gupta et al. "Improved Modeling of 3D Shapes with Multi-view Depth Maps". In: *arXiv preprint arXiv:2009.03298* (2020).

[Gup+21] Kamal Gupta et al. "LayoutTransformer: Layout Generation and Completion with Self-attention". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 1004–1014.

[GSH19] Tanmay Gupta, Alexander Schwing, and Derek Hoiem. "ViCo: Word Embeddings from Visual Co-occurrences". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 7425–7434.

[HT17] Serhii Havrylov and Ivan Titov. "Emergence of language with multi-agent games: Learning to communicate with sequences of symbols". In: *arXiv preprint arXiv:1705.11192* (2017).

[He+16] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[He+20] Kaiming He et al. "Momentum contrast for unsupervised visual representation learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 9729–9738.

[HHW19] Tobias Hinz, Stefan Heinrich, and Stefan Wermter. "Generating Multiple Objects at Spatially Distinct Locations". In: *CoRR* abs/1901.00686 (2019). arXiv: 1901.00686. URL: http://arxiv.org/abs/1901.00686.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[Hol+19] Ari Holtzman et al. "The curious case of neural text degeneration". In: *arXiv preprint arXiv:1904.09751* (2019).

[Hon+18] Seunghoon Hong et al. "Inferring semantic layout for hierarchical text-to-image synthesis". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 7986–7994.

[Iso+16] Phillip Isola et al. "Image-to-Image Translation with Conditional Adversarial Networks". In: *arxiv* (2016).

[JGP16] Eric Jang, Shixiang Gu, and Ben Poole. "Categorical reparameterization with gumbel-softmax". In: *arXiv preprint arXiv:1611.01144* (2016).

[JGF18] 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*. 2018, pp. 1219–1228.

[Jor+16] Emilio Jorge et al. "Learning to play guess who? and inventing a grounded language as a consequence". In: *arXiv preprint arXiv:1611.03218* (2016).

[Jun+13] Mayank Juneja et al. “Blocks that shout: Distinctive parts for scene classification”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 923–930.

[JT05] Frederic Jurie and Bill Triggs. “Creating efficient codebooks for visual recognition”. In: *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*. Vol. 1. IEEE. 2005, pp. 604–610.

[Jyo+19] Akash Abdu Jyothi et al. “Layoutvae: Stochastic scene layout generation from a label set”. In: *arXiv preprint arXiv:1907.10719* (2019).

[KLA19] Tero Karras, Samuli Laine, and Timo Aila. “A style-based generator architecture for generative adversarial networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4401–4410.

[Kar+17] Tero Karras et al. “Progressive growing of gans for improved quality, stability, and variation”. In: *arXiv preprint arXiv:1710.10196* (2017).

[Kaz+14] Sahar Kazemzadeh et al. “Referitgame: Referring to objects in photographs of natural scenes”. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 787–798.

[KB20] Bence Keresztury and Elia Bruni. “Compositional properties of emergent languages in deep learning”. In: *arXiv preprint arXiv:2001.08618* (2020).

[KHB18] Salman H Khan, Munawar Hayat, and Nick Barnes. “Adversarial Training of Variational Auto-encoders for High Fidelity Image Generation”. In: *arXiv preprint arXiv:1804.10323* (2018).

[KM18] Hyunjik Kim and Andriy Mnih. “Disentangling by factorising”. In: *arXiv preprint arXiv:1802.05983* (2018).

[KB14] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[KW13a] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2013. arXiv: 1312.6114 [stat.ML].

[KW13b] Diederik P Kingma and Max Welling. “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114* (2013).

[Kin+16] Diederik P Kingma et al. “Improved variational inference with inverse autoregressive flow”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 4743–4751.

[Kin+14] Durk P Kingma et al. “Semi-supervised learning with deep generative models”. In: *Advances in neural information processing systems*. 2014, pp. 3581–3589.

[Kir02] Simon Kirby. “Natural language from artificial life”. In: *Artificial life* 8.2 (2002), pp. 185–215.

[Kri+18] Ranjay Krishna et al. “Referring relationships”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 6867–6876.

[KH+09] Alex Krizhevsky, Geoffrey Hinton, et al. *Learning multiple layers of features from tiny images*. Tech. rep. Citeseer, 2009.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

[Lar+15] Anders Boesen Lindbo Larsen et al. "Autoencoding beyond pixels using a learned similarity metric". In: *arXiv preprint arXiv:1512.09300* (2015).

[LPB16] Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. "Multi-agent cooperation and the emergence of (natural) language". In: *arXiv preprint arXiv:1612.07182* (2016).

[Laz+18] Angeliki Lazaridou et al. "Emergence of linguistic communication from referential games with symbolic and pixel input". In: *arXiv preprint arXiv:1804.03984* (2018).

[Lee+18] Donghoon Lee et al. "Context-Aware Synthesis and Placement of Object Instances". In: *CoRR* abs/1812.02350 (2018). arXiv: 1812.02350. URL: http://arxiv.org/abs/1812.02350.

[Lee+17] Jason Lee et al. "Emergent translation in multi-agent communication". In: *arXiv preprint arXiv:1710.06922* (2017).

[Lew69] David Lewis. *Convention: A philosophical study*. John Wiley & Sons, 1969.

[Li+19a] Boren Li et al. "Seq-sg2sl: Inferring semantic layout from scene graph through sequence to sequence learning". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 7435–7443.

[Li+19b] Jianan Li et al. "LayoutGAN: Generating Graphic Layouts with Wireframe Discriminators". In: *arXiv preprint arXiv:1901.06767* (2019).

[Li+19c] Manyi Li et al. "Grains: Generative recursive autoencoders for indoor scenes". In: *ACM Transactions on Graphics (TOG)* 38.2 (2019), pp. 1–16.

[Li+19d] Wenbo Li et al. "Object-driven Text-to-Image Synthesis via Adversarial Training". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 12174–12182.

[Lim+18] Isaak Lim et al. "A simple approach to intrinsic correspondence learning on unstructured 3d meshes". In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. 2018, pp. 0–0.

[Lin+14] Tsung-Yi Lin et al. "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[Liu+18] Thomas F. Liu et al. "Learning Design Semantics for Mobile Apps". In: *The 31st Annual ACM Symposium on User Interface Software and Technology*. UIST '18. Berlin, Germany: ACM, 2018, pp. 569–579. ISBN: 978-1-4503-5948-1. DOI: 10.1145/3242587.3242650. URL: http://doi.acm.org/10.1145/3242587.3242650.

[Low04] David G Lowe. “Distinctive image features from scale-invariant keypoints”. In: *International journal of computer vision* 60.2 (2004), pp. 91–110.

[Low+19] Ryan Lowe et al. “On the pitfalls of measuring emergent communication”. In: *arXiv preprint arXiv:1903.05168* (2019).

[MMT16] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. “The concrete distribution: A continuous relaxation of discrete random variables”. In: *arXiv preprint arXiv:1611.00712* (2016).

[MRC20] Dipu Manandhar, Dan Ruta, and John Collomosse. “Learning Structural Similarity of User Interface Layouts using Graph Networks”. In: *ECCV*. 2020.

[Mat+17] Loïc Matthey et al. “beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework”. In: *ICLR 2017*. 2017.

[MH19] Daniela Mihai and Jonathon Hare. “Avoiding hashing and encouraging visual semantics in referential emergent language games”. In: *arXiv preprint arXiv:1911.05546* (2019).

[MH21] Daniela Mihai and Jonathon Hare. “The emergence of visual semantics through communication games”. In: *arXiv preprint arXiv:2101.10253* (2021).

[MJB16] Tomas Mikolov, Armand Joulin, and Marco Baroni. “A roadmap towards machine intelligence”. In: *International Conference on Intelligent Text Processing and Computational Linguistics*. Springer. 2016, pp. 29–61.

[Mik+13] Tomas Mikolov et al. “Efficient estimation of word representations in vector space”. In: *arXiv preprint arXiv:1301.3781* (2013).

[Mil+20] Ben Mildenhall et al. “Nerf: Representing scenes as neural radiance fields for view synthesis”. In: *European conference on computer vision*. Springer. 2020, pp. 405–421.

[MM20] 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*. 2020, pp. 6707–6717.

[Mo+19] Kaichun Mo et al. “Structurenet: Hierarchical graph networks for 3d shape generation”. In: *arXiv preprint arXiv:1908.00575* (2019).

[MA18] Igor Mordatch and Pieter Abbeel. “Emergence of grounded compositional language in multi-agent populations”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 2018.

[Nas+21] Charlie Nash et al. “Generating Images with Sparse Representations”. In: *arXiv preprint arXiv:2103.03841* (2021).

[NF16] Mehdi Noroozi and Paolo Favaro. “Unsupervised learning of visual representations by solving jigsaw puzzles”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 69–84.

[OK16] Aäron van den Oord and Nal Kalchbrenner. “Pixel RNN”. In: (2016).

[OKK16] Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. "Pixel Recurrent Neural Networks". In: *CoRR* abs/1601.06759 (2016). arXiv: 1601.06759. URL: http://arxiv.org/abs/1601.06759.

[OLV18] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. "Representation learning with contrastive predictive coding". In: *arXiv preprint arXiv:1807.03748* (2018).

[OVK17] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. "Neural discrete representation learning". In: *arXiv preprint arXiv:1711.00937* (2017).

[Par+19a] Jeong Joon Park et al. "DeepSDF: Learning continuous signed distance functions for shape representation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 165–174.

[Par+19b] Taesung Park et al. "GauGAN: semantic image synthesis with spatially adaptive normalization". In: *ACM SIGGRAPH 2019 Real-Time Live!* ACM. 2019, p. 2.

[Pat+16] Deepak Pathak et al. "Context encoders: Feature learning by inpainting". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2536–2544.

[Pat+17] Deepak Pathak et al. "Learning Features by Watching Objects Move". In: *CVPR*. 2017.

[Pat+19] Akshay Gadi Patil et al. "Read: Recursive autoencoders for document layout generation". In: *arXiv preprint arXiv:1909.00302* (2019).

[Pin03] Steven Pinker. *The language instinct: How the mind creates language*. Penguin UK, 2003.

[QT09] Ariadna Quattoni and Antonio Torralba. "Recognizing indoor scenes". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2009, pp. 413–420.

[RMC15] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: *CoRR* abs/1511.06434 (2015). arXiv: 1511.06434. URL: http://arxiv.org/abs/1511.06434.

[Rad+18] Alec Radford et al. "Improving language understanding by generative pre-training". In: (2018).

[Ram+21] Aditya Ramesh et al. "Zero-shot text-to-image generation". In: *arXiv preprint arXiv:2102.12092* (2021).

[ROV19] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. "Generating diverse high-fidelity images with vq-vae-2". In: *arXiv preprint arXiv:1906.00446* (2019).

[Ree+16] Scott Reed et al. "Generative adversarial text to image synthesis". In: *arXiv preprint arXiv:1605.05396* (2016).

[RWL19] 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*. 2019, pp. 6182–6190.

[Rol16] Jason Tyler Rolfe. "Discrete variational autoencoders". In: *arXiv preprint arXiv:1609.02200* (2016).

[Sal+17a] Tim Salimans et al. "PixelCNN++: A PixelCNN Implementation with Discretized Logistic Mixture Likelihood and Other Modifications". In: *ICLR*. 2017.

[Sal+17b] Tim Salimans et al. "Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications". In: *arXiv preprint arXiv:1701.05517* (2017).

[Sal+17c] Tim Salimans et al. "Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications, 2017". In: *arXiv preprint arXiv:1701.05517* (2017).

[Sel+17] Ramprasaath R Selvaraju et al. "Grad-cam: Visual explanations from deep networks via gradient-based localization". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 618–626.

[SG16] Abhinav Shrivastava and Abhinav Gupta. "Contextual priming and feedback for faster r-cnn". In: *European Conference on Computer Vision*. Springer. 2016, pp. 330–348.

[SZ14] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[SGE12a] Saurabh Singh, Abhinav Gupta, and Alexei A Efros. "Unsupervised discovery of mid-level discriminative patches". In: *European Conference on Computer Vision*. Springer. 2012, pp. 73–86.

[SGE12b] Saurabh Singh, Abhinav Gupta, and Alexei A. Efros. "Unsupervised Discovery of Mid-level Discriminative Patches". In: *European Conference on Computer Vision*. 2012. arXiv: 1205.3137 [cs.CV]. URL: http://arxiv.org/abs/1205.3137.

[SG05] Linda Smith and Michael Gasser. "The development of embodied cognition: Six lessons from babies". In: *Artificial life* 11.1-2 (2005), pp. 13–29.

[Smo88] Paul Smolensky. *Connectionism, constituency, and the language of thought*. University of Colorado at Boulder, 1988.

[Soh16] 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*. 2016, pp. 1857–1865.

[SLY15] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. "Learning structured output representation using deep conditional generative models". In: *Advances in neural information processing systems*. 2015, pp. 3483–3491.

[Spi+17] Matthew Spike et al. “Minimal requirements for the emergence of learned signaling”. In: *Cognitive science* 41.3 (2017), pp. 623–658.

[Ste05] Luc Steels. “What triggers the emergence of grammar?” In: *Society for the Study of Artificial Intelligence and Simulation of Behaviour* (2005).

[STN94] Volker Steinbiss, Bach-Hiep Tran, and Hermann Ney. “Improvements in beam search”. In: *Third International Conference on Spoken Language Processing*. 1994.

[SSF16] Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. “Learning multiagent communication with backpropagation”. In: *arXiv preprint arXiv:1605.07736* (2016).

[Sun+17a] Chen Sun et al. “Revisiting Unreasonable Effectiveness of Data in Deep Learning Era”. In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 843–852.

[Sun+17b] Minhyuk Sung et al. “Complementme: weakly-supervised component suggestions for 3D modeling”. In: *ACM Transactions on Graphics (TOG)* 36.6 (2017), pp. 1–12.

[Sze+16] Christian Szegedy et al. “Rethinking the inception architecture for computer vision”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826.

[] *The Level Design*. https://book.leveldesignbook.com/process/blockout. Accessed: 2021-11-24.

[TS01] Antonio Torralba and Pawan Sinha. “Statistical context priming for object detection”. In: *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*. Vol. 1. IEEE. 2001, pp. 763–770.

[Vas+17] Ashish Vaswani et al. “Attention is all you need”. In: *arXiv preprint arXiv:1706.03762* (2017).

[VBK15] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. “Order matters: Sequence to sequence for sets”. In: *arXiv preprint arXiv:1511.06391* (2015).

[VPT16] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. “Generating videos with scene dynamics”. In: *Advances in neural information processing systems*. 2016, pp. 613–621.

[Wag+03] Kyle Wagner et al. “Progress in the simulation of emergent communication and language”. In: *Adaptive Behavior* 11.1 (2003), pp. 37–69.

[Wan+18a] Kai Wang et al. “Deep convolutional priors for indoor scene synthesis”. In: *ACM Transactions on Graphics (TOG)* 37.4 (2018), p. 70.

[Wan+19] Kai Wang et al. “Planit: Planning and instantiating indoor scenes with relation graph and spatial prior networks”. In: *ACM Transactions on Graphics (TOG)* 38.4 (2019), pp. 1–15.

[Wan+18b] Ting-Chun Wang et al. “High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

[WG15] Xiaolong Wang and Abhinav Gupta. “Unsupervised learning of visual representations using videos”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 2794–2802.

[WHG17] 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.

[Wit53] Ludwig Wittgenstein. *Philosophical investigations*. John Wiley & Sons, 1953.

[WTK17] Jiajun Wu, Joshua B Tenenbaum, and Pushmeet Kohli. “Neural Scene De-rendering”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[Wu+16] Jiajun Wu et al. “Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling”. In: *Advances in neural information processing systems*. 2016, pp. 82–90.

[Wu+17] Jiajun Wu et al. “Learning to See Physics via Visual De-animation”. In: *Advances in Neural Information Processing Systems*. 2017.

[Wu+20] Rundi Wu et al. “PQ-NET: A generative part seq2seq network for 3D shapes”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 829–838.

[Yan+19] Guandao Yang et al. “PointFlow: 3D point cloud generation with continuous normalizing flows”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 4541–4550.

[Yan+20] Kaiyu Yang et al. “Towards fairer datasets: Filtering and balancing the distribution of the people subtree in the imagenet hierarchy”. In: *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. 2020, pp. 547–558.

[Yan+21] Kaiyu Yang et al. “A Study of Face Obfuscation in ImageNet”. In: *arXiv preprint arXiv:2103.06191* (2021).

[Yan+17] Xiao Yang et al. “Learning to Extract Semantic Structure from Documents Using Multimodal Fully Convolutional Neural Network”. In: *CoRR* abs/1706.02337 (2017). arXiv: 1706.02337. URL: http://arxiv.org/abs/1706.02337.

[Yu+19] Fenggen Yu et al. “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*. 2019, pp. 9491–9500.

[YKH10] Cem Yuksel, John Keyser, and Donald H House. “Mesh colors”. In: *ACM Transactions on Graphics (TOG)* 29.2 (2010), pp. 1–11.

[Zbo+21] Jure Zbontar et al. “Barlow twins: Self-supervised learning via redundancy reduction”. In: *arXiv preprint arXiv:2103.03230* (2021).

[Zha+17] Han Zhang et al. “Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 5907–5915.

[Zha+18] Han Zhang et al. “Self-Attention Generative Adversarial Networks”. In: *arXiv preprint arXiv:1805.08318* (2018).

[Zha+19] Bo Zhao et al. “Image Generation from Layout”. In: *CVPR*. 2019.

[Zhe+19] Xinru Zheng et al. “Content-aware generative modeling of graphic design layouts”. In: *ACM Transactions on Graphics (TOG)* 38.4 (2019), pp. 1–15.

[ZTY19] 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)*. IEEE. 2019, pp. 1015–1022.

[Zho+17] Bolei Zhou et al. “Places: A 10 million image database for scene recognition”. In: *IEEE transactions on pattern analysis and machine intelligence* 40.6 (2017), pp. 1452–1464.

[Zhu+17] Jun-Yan Zhu et al. “Unpaired image-to-image translation using cycle-consistent adversarial networks”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2223–2232.

[Zou+17] Chuhang Zou et al. “3d-prnn: Generating shape primitives with recurrent neural networks”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 900–909.