Disentangled Visual Generative Models

By

Dave Epstein

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Alexei A. Efros, Chair
Professor Angjoo Kanazawa
Professor Trevor Darrell
Doctor Ben Poole

Spring 2024

Disentangled Visual Generative Models

Abstract

Disentangled Visual Generative Models

by

Dave Epstein

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Alexei A. Efros, Chair

Generative modeling promises an elegant solution to learning about high-dimensional data distributions such as images and videos — but how can we expose and utilize the rich structure these models discover? Rather than just drawing new samples, how can an agent actually harness $p(x)$ as a source of knowledge about how our world works? This thesis explores scalable inductive biases that unlock a generative model's disentangled understanding of visual data, enabling much richer interaction and control as a result.

First, I propose a representation of scenes as collections of feature "blobs", where a generative adversarial network (GAN) learns – without any labels – to bind each blob to a different object in the images it creates. This allows GANs to more gracefully model compositional scenes, in contrast to typical unconditional models which are constrained to highly-aligned single-object data. The trained model's representation can easily be modified to counterfactually manipulate objects in both generated and real images.

Next, I consider methods that do not impose bottlenecks on architectures during training, facilitating their application to more diverse, uncurated data. I show that the internals of diffusion models can be used to meaningfully guide generation of new samples, without any further fine-tuning or supervision. Energy functions derived from a small set of primitive properties of denoiser activations can be combined to impose arbitrarily complex conditions on the iterative diffusion sampling procedure. This allows for control over attributes such as the position, shape, size, and appearance of any concept that can be described in text.

I also demonstrate that the distribution learned by a text-to-image model can be distilled to generate compositional 3D scenes. Predominant approaches focus on creating 3D objects in isolation rather than scenes with several entities interacting. I propose an architecture that, when optimized so its outputs are on-manifold for the image generator, creates 3D scenes decomposed into the objects they contain. This provides evidence that scale alone suffices for a model to infer the actual 3D structure latent to a world it observes only through 2D images.

Finally, I conclude with a perspective on the interplay between emergence, control, interpretability, and scale, and humbly attempt to relate these themes to the pursuit of intelligence.

To the family that we choose, and that chooses us back.

Contents

List of Figures

List of Tables

Acknowledgments

> Mais c'est cela l'amour, tout donner, tout sacrifier sans espoir de retour.
>
> ---
>
> Les justes
> *Albert Camus*

As I began to search for the right way to open this section (by far the most important of the thesis) I realized that four years ago, to the day, I accepted Alyosha's offer to join Berkeley. Much has changed in these past years. For one, most of the reasons for which I chose to attend graduate school, to pursue a career in research – much less in AI – do not seem to hold true in my heart today. Naturally, this has been a source of great existential consternation, but I'm somehow still extremely glad I went down this path. Perhaps this is due to the incredible constellation of people that this journey has granted me the profound privilege of meeting and learning from.

First, to Alyosha. Your seductive photo of a sunset from the hills was an extremely effective recruitment tactic. The hikes that followed in those same hills – tokens of sanity in an otherwise insane year of remoteness – not only shaped the questions I sought out to answer in this document, but forever changed how I view the enterprise of conducting research. For your patience and trust as I indulged my inner flâneur, and for the cosmic rays of creative entropy our conversations spawned along the way, I am deeply grateful.

To members of my committee, Trevor Darrell, Angjoo Kanazawa, and Ben Poole. The first mistake I ever made in Berkeley was turning down Trevor's generous offer of a bundle of N95 masks when I visited in early 2020; Trevor continued to be supportive and considerate despite this fact. To Angjoo, for forgiving me for fattening up her cats on various occasions, and striking an artful balance between treating me as an equal and offering guidance in moments of need. To Ben, for an extremely high signal-to-noise ratio and an unwaveringly calm presence, for your generosity with your time and energy. To Carl Vondrick, who took a naïf 19-year-old me under his wing and dedicated an illogical portion of his calendar to helping me learn what research is. I am forever indebted to Carl, without whom I would have never imagined a future in science. To Paul Blaer, for picking me as his twelfth teaching assistant for a data structures class seven years ago.

To the collaborators I've been lucky to have: Richard Zhang and Eli Shechtman, for not caring that it took me a month to train my first GAN because I was using AdamW instead of Adam, and still letting me spread my wings. To submitting a paper at 11:59:55 AM. To Ben Mildenhall, for helping me find steady footing in three dimensions. To Taesung Park, not for our fateful chat in Alyosha's office that led to blobs, or for last-minute precision-recall curves, but for the discussions we have shared since, for effortlessly demonstrating how to be both a good researcher and a good person — to 10cm. To Aleks Hołyński, for enabling me to roam while shepherding me forth all the while, for the best potato vodka I've ever had (though no kabanos). To Allan Jabri, habibi. Despite slight differences in attachment style, I found a kindred spirit in Allan, who imbued me with courage and confidence while bringing me back down to earth when needed. The course I have charted in research and in life has been invariably affected by your mentorship and care throughout the years – thank you.

To Berkeley friends – pithy one-liners do not begin to describe the immeasurable impact your support has had on me. To Laura Smith, for badminton, Basque cheesecake, COGSCI C100, spanakopita at Nostos (RIP), and not letting each other drop out. To Sal Fu, for okra salad and chance encounters. To Vickie Ye, for commiseration over corn cherry scones. To Yu Sun, for expert driving skills, Brahms, maotai, and our long-distance relationship (Berkeley-Palo Alto). To Ilija Radosavovic, for campus walks and measured wisdom. To Tim Brooks and Bill Peebles, for clearing the grasses and showing me where to step. To Shiry Ginosar, for Thanksgiving dinner and letting me taste-test ice cream on her children. To Georgios Pavlakos, for kebabs at Ohlone Park, before we knew what we were getting ourselves into. To Yossi Gandelsman, for cardamom coffee and our shared experiences. To Assaf Shocher, for making me not be the person with the driest sense of humor in our group. To Toru Lin, for boba and gossip. To Purvi Goel, for Hopkins Street.

To friends from other chapters in life: Georgia Essig and Niki Konforti, for predicting all the mistakes I'd make, letting me make them, and waiting for me on the other side – to growth. To Minyoung Huh, for Busker Busker and interview prep. To Mert Uşşaklı, for a carefree summer in Seattle and a magical week in Istanbul. To Mia Chiquier, for lambig, Emma Peters, and grisaille. To Kris Papa, the closest I will ever get to having a true lifelong friend, for Fanta Shokata and Kith notebooks. To Lucy Chai, for moral support, manuscript revisions, and San Diego. To Dan Bastian, Brett Kozinn, Dallas Shatel, Pete Campora, Ed Fisher, Henrik Liu, Paul Dutton, Bear Busby, Jeff Krismer, David Lozano, Cogan Bishop, Dan Ohlemiller, Ryan Brunner, Sam Callahan, Huan Nguyen, and Brad Bogdan: For teaching me something new about friendship, for impeccable taste in rum and mezcal, for opening up your homes to me. See you in Oaxaca.

Chapter 1

Introduction

> El mundo era tan reciente, que muchas cosas carecían de nombre, y para mencionarlas había que señalarlas con el dedo.
>
> Cien años de soledad
> *Gabriel Garcia Márquez*

Any intelligent agent must have the ability to perceive the world around it. No event in the history of our planet better demonstrates this fact than the Cambrian explosion [101, 85]. Around half a billion years ago (at time of writing), organisms suddenly developed a primitive precursor to what we would recognize today as eyes. Equipped with such a powerful tool, these organisms were fitter than others and thus survived. How can we teach a machine to see?

1.1 Learning without labels

Progress in deep learning in the past five or ten years has all yielded the same simple message – the more data your algorithm can process, the better [171]. To process more data, the number of parameters in the algorithm must grow [71, 59]. It follows that we require a method that can learn well at scale, as well as a massive amount of data to use as input for this method. It is infeasible to manually annotate many billions of images and videos with descriptions of what we'd like an algorithm to learn from each example. Therefore, I restrict my attention to methods that do not assume access to any labels and only operate on the data itself.

Much of the modern work in this direction can be categorized into two classes: "self-supervised" and "generative" approaches. In self-supervision, we withhold some aspect of the input data and task our model with predicting it (Figure 1.1), hoping that the difficulty of the prediction leads the model to learn a representation of the world that we can later use to

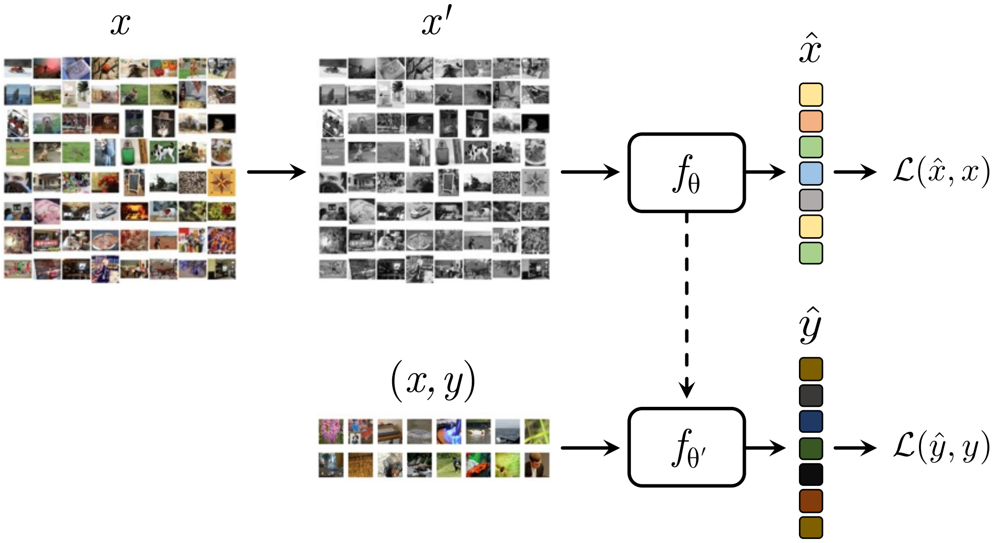


Figure 1.1: **Self-supervision.** Given unlabeled data x, self-supervision gives a network some portion x' of the input and tasks it with predicting the other. For example, we may ask a network to predict the missing color channels of a grayscale image (as shown), or remove certain patches of the image and ask the model to predict them [29, 47], yielding a reconstruction $\hat{x}$. The trained network weights θ can then be used as an initialization for training a second network to solve a task of interest, *e.g.* classification.

solve a variety of problems. In generative modeling, we ask a network to learn the underlying distribution of our dataset, $p(x)$. We do this, for example, by training it to maximize the likelihood assigned to the samples of data it observes. We can then use our model to draw new samples from the same distribution as our training data, and hope that the learned distribution can be employed to solve downstream tasks.

1.2 Demonstrating and accessing knowledge

In either case, once we have devised a recipe for training a neural network on unlabeled data, we must figure out how to apply this model to a concrete problem of interest – that is, how our model might demonstrate what it has learned. This step is inevitable, as our model's representation must be bound to variables that we are able to interpret.

In the case of self-supervision, the representation can either be used to induce a similarity metric [170, 36] — expressing its knowledge through ranking and retrieval — or further trained in some way with a smaller set of labeled data [32, 31].[1] Both paradigms are severely

[1] Please note that I refer to my own previous projects here not to inflate my citation count, but to avoid criticizing the reasonable work of other researchers.

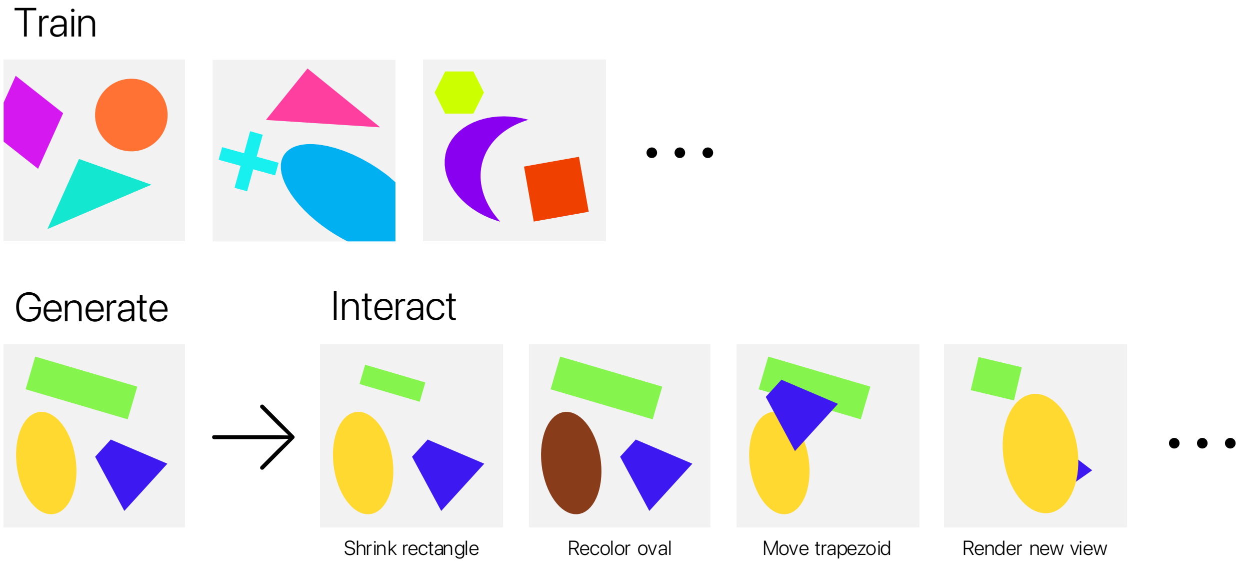


Figure 1.2: **Knowledge $\iff$ control.** Manipulating one and only one aspect of a generated image provides a simple, intuitive way to show that the model has acquired a rich understanding of the data distribution. This can be thought of as a sort of counterfactual proof of knowledge.

limited. Fine-tuning requires the collection of ground truth for every task we wish to solve; this not only makes evaluating the representation itself more complicated, but obfuscates the value of obtaining a generic pretrained representation in the first place. Retrieval necessitates the maintenance of a large bank of potential answers, limiting expressivity (both interpolation and extrapolation are impossible by construction) and introducing complications such as approximate nearest-neighbor algorithms.

Generative modeling, however, presents a unique opportunity for “proof of knowledge”. The fundamental distinction between this class of models and self-supervised ones is that samples from a generative model are themselves interpretable, so we can perturb the model’s internals and inspect how the output changes. This enables a form of counterfactual interventions, wherein we can claim a model has discovered structure if it is able to, *e.g.*, alter an image only along a certain axis (Figure 1.2), leaving all else untouched. This highlights a deep duality between a model’s knowledge and its ability to control its outputs. In theory, this direction requires no additional labels or supervision in order to make use of our trained model, though in practice, many approaches do use auxiliary classifiers to help detect meaningful directions in the learned latent space [21, 67]. Of course, it is exactly this sort of reliance on labels that we seek to eschew in the first place.

1.3 Contributions

In this thesis, I explore different approaches to accessing the knowledge learned by generative models. In addition to scaling elegantly, these approaches provide novel forms of control over the outputs of generative models trained on visual data. In Chapter 2, I propose a disentangled representation of scenes as collections of feature "blobs", where a generative adversarial network learns – without any labels – to bind each blob to a different object in the images it creates. Next, I investigate methods that do not impose bottlenecks on architectures during training. In Chapter 3, I show that the internals of diffusion models can be used to meaningfully guide sampling of new data at inference. In Chapter 4, I demonstrate that the distribution learned by these models can be distilled to generate compositional 3D scenes.

Chapter 2 is adapted from *BlobGAN: Spatially Disentangled Scene Representations* at ECCV 2022, joint work with Taesung Park, Richard Zhang, Eli Shechtman, and Alexei A. Efros. [33]

Chapter 3 is adapted from *Diffusion Self-Guidance for Controllable Image Generation* at NeurIPS 2023, joint work with Allan Jabri, Ben Poole, Alexei A. Efros, and Aleksander Holynski. [34]

Chapter 4 is adapted from *Disentangled 3D Scene Generation with Layout Learning* at ICML 2024, joint work with Ben Poole, Ben Mildenhall, Alexei A. Efros, and Aleksander Holynski. [35]

Chapter 2

A Representation for Generative Object Discovery

We begin by proposing an unsupervised, mid-level representation for a generative model of scenes. The representation is mid-level in that it is neither per-pixel nor per-image; rather, scenes are modeled as a collection of spatial, depth-ordered “blobs” of features. Blobs are differentiably placed onto a feature grid that is decoded into an image by a generative adversarial network. Due to the spatial uniformity of blobs and the locality inherent to convolution, our network learns to associate different blobs with different entities in a scene and to arrange these blobs to capture scene layout. We demonstrate this emergent behavior by showing that, despite training without any supervision, our method enables applications such as easy manipulation of objects within a scene (*e.g.* moving, removing, and restyling furniture), creation of feasible scenes given constraints (*e.g.* plausible rooms with drawers at a particular location), and parsing of real-world images into constituent parts. On a challenging multi-category dataset of indoor scenes, BlobGAN outperforms StyleGAN2 in image quality as measured by FID.

2.1 Introduction

The visual world is incredibly rich. It is so much more than the typical ImageNet-style photos of solitary, centered objects (cars, cats, birds, faces, *etc.*), which are the mainstays of most current paper result sections. Indeed, it was long clear, both in human vision [14, 57] and in computer vision [189, 119, 121, 60, 43], that understanding and modeling objects within the context of a *scene* is of the utmost importance. Visual artists have understood this for centuries, first by discovering and following the rules of scene formation during the Renaissance, and then by expertly breaking such rules in the 20th century (cf. the surrealists including Magritte, Ernst, and Dalí).

However, in the current deep learning era, scene modeling for both analysis and synthesis tasks has largely taken a back seat. Images of scenes are either represented in a top-down

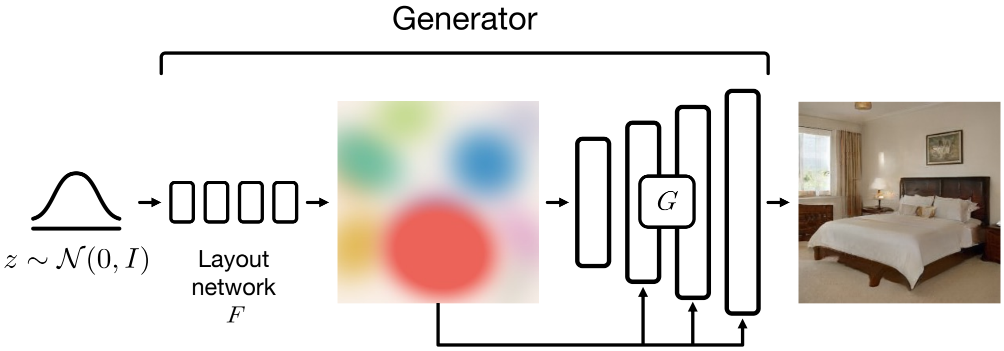


Figure 2.1: **BlobGAN:** In our generator, random noise is mapped by the layout network F to blob parameters. Blobs output by F are *splatted* spatially onto corresponding locations in the feature grid, used both as initial input and as spatially-adaptive modulation for the convolutional decoder G. Our blob representation automatically serves as a strong mid-level generative representation for scenes, discovering objects and their layouts.

fashion, no different from objects – *i.e.* for GANs or image classifiers, scene classes such as "bedrooms" or "kitchens" are represented the same way as object classes, such as "beds" or "chairs". Or, scenes are modeled in a bottom-up way by semantic labeling of each image pixel, *e.g.*, semantic segmentation, pix2pix [63], SPADE [123], *etc.* Both paths seem unsatisfactory because neither can provide easy ways of reasoning about parts of the scene as *entities*. The scene parts are either baked into a single entangled latent vector (top-down), or need to be grouped together from individual pixel labels (bottom-up).

In this paper, we propose an *unsupervised mid-level representation* for a generative model of scenes. The representation is mid-level in that it is neither per-pixel nor per-image; rather, scenes are modeled as a collection of spatial, depth-ordered Gaussian "blobs". This collection of blobs provides a bottleneck in the generative architecture, as shown in Figure 2.1, forcing each blob to correspond to a specific object in the scene and thus causing a spatially disentangled representation to emerge. This representation allows us to perform a number of scene editing tasks (see Figure 2.3) previously only achievable with extensive semantic supervision, if at all.

2.2 Related work

Mid-level scene representations. Work on mid-level scene representations can be traced back to the 1970s, to the seminal papers of Yakimovsky and Feldman [189] and Ohta et al [119], which already contained many key ideas including joint bottom-up segmentation and top-down reasoning. Other important developments were the line of work on normalized-cuts segmentation [158, 198, 45] and qualitative 3D scene interpretation [60, 49, 161, 43] in the early 2000s. But most relevant to the present manuscript is the classic *Blobworld* work of Carson et al. [20], a region-based image retrieval system, with each image represented by a mixture-of-Gaussian blobs. Our model could be considered a generative version of this representation, except we also encode the depth ordering of the blobs.

Scene analysis by synthesis. The idea of modeling a complex visual scene by trying to generate it has been attempted a number of times in the past. Early methods, such as [176, 169, 174], introduced key ideas but were limited by the generative models of the time. To address this, several approaches tried non-parametric generation [100, 148, 62], with Scene Collaging [62] the most valiant attempt, showing layered scene representations despite very heavy computational burden. With the advancement of deep generative models, parametric analysis-by-synthesis techniques are having a renaissance, with some top-down [192, 120, 195] as well as bottom-up [126, 47] techniques.

Conditional image generation. Conditional GANs [208, 61, 181], such as image-to-image translation setups [63], predict an image from a predefined representation, *e.g.* semantic segmentation maps [123, 91], object-attribute graphs [69, 12], text [201, 138, 113, 135, 142, 137], pose [87, 154, 3], and keypoints [18]. Other setups include using perceptual losses [24], implicit likelihood estimation [89], and more recently, diffusion models [105, 150]. [160, 103, 102, 159, 180, 48] explore related intermediate representations to help generation (mostly of humans or objects) but none provide the ability to generate and manipulate high-quality scene images of our method.

Unconditional generation and disentanglement. Rather than use explicit conditioning, it is possible to learn an image "manifold" with a generative model such as a VAE [80, 54] or GAN [41] and explore emergent capabilities. GANs have improved in image quality [133, 28, 200, 17, 76, 72, 74, 73] and are our focus. Directions of variation naturally emerge in the latent space and can be discovered when guided by geometry/color changes [67], language or attributes [67, 125, 133, 157, 2, 186], cognitive signals [40], or in an unsupervised manner [46, 156, 127]. Discovering disentangled representations remains a challenging open problem [97, 46, 156, 127]. To date, most successful applications have been on data of objects, *e.g.* faces and cars, or changing textures for scenes [124, 97, 46, 156, 127]. Similar to us, an active line of work explores adding 3D inductive biases [115, 110, 111, 124, 97, 46, 156, 127], but individual object manipulation has largely focused on simple diagnostic scenes [70]. Alternatively, the internal units of a pretrained GAN offer finer spatial control, with certain units naturally correlating with object classes [9, 10, 190]. The internal compositionality of GANs can be leveraged to harmonize images [26, 21] or perform a limited set of edits on objects in a scene [9, 199, 206]. Crucially, while these works require semantic supervision to identify units and regions, our work uses a representation where these factors naturally emerge.

2.3 Method

Our method aims to learn a representation of scenes as spatial maps of blobs through the generative process. As shown in Figure 2.1, a layout network maps from random noise to a

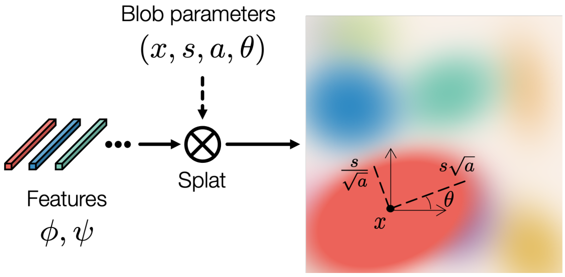


Figure 2.2: **BlobGAN architecture:** Our elliptical blobs β are parametrized by centroid x, scale s, aspect ratio a, and angle θ. We composite multiple blobs with alpha values that smoothly decay from blob centers. The features ϕ or ψ are splatted on their corresponding blobs and passed to the decoder.

set of blob parameters. Then, blobs are differentiably splatted onto a spatial grid – a "blob map" – which a StyleGAN2-like decoder [75] converts into an image. Finally, the blob map is used to modulate the decoder We train our model in an adversarial framework with an unmodified discriminator [74]. Interestingly, even without explicit labels, our model learns to decompose scenes into entities and their layouts.

Our generator model is largely divided into two parts. First, we apply an 8-layer MLP F to map random noise $z \in \mathbb{R}^{d_{\text{noise}}} \sim \mathcal{N}(0, I_d)$ to a collection of blobs parameterized by $\boldsymbol{\beta} = \{\beta_i\}_{i=1}^{k}$ which are splatted onto a spatial $H \times W \times d$ feature grid in a differentiable manner. This process is visualized in Figure 2.2. The feature grid is then passed to a convolutional decoder G to produce final output images. In the remainder of this section, we describe the design of our representation as well as its implementation in detail.

2.3.1 From noise to blobs as layout

We map from random Gaussian noise to distributions of blobs with an MLP F with dimension d_{hidden}. The last layer of F is decoded into a sequence of blob properties $\boldsymbol{\beta}$. We opt for a simple yet effective parametrization of blobs, representing them as ellipses by their center coordinates $x \in [0,1]^2$, scale $s \in \mathbb{R}$, aspect ratio $a \in \mathbb{R}$, and rotation angle $\theta \in [-\pi, \pi]$. Each blob is also associated with a structure feature $\phi \in \mathbb{R}^{d_{\text{in}}}$ and a style feature $\psi \in \mathbb{R}^{d_{\text{style}}}$. Altogether, our blob representation is:

$$\beta \in \mathbb{R}^{2+1+1+1+d_{\text{in}}+d_{\text{style}}} \triangleq (x, s, a, \theta, \phi, \psi)$$

Next, we transform the blob parameters to a 2D feature grid by populating the ellipse specified by β with the feature vectors ϕ and ψ. We do this differentiably by assigning an opacity and spatial falloff to each blob. Specifically, we calculate a grid $\boldsymbol{\alpha} \in [0,1]^{H \times W \times k}$ which indicates each blob's opacity value at each location. We then use these opacity maps to *splat* the features ϕ, ψ at various resolutions, using a single broadcasted matrix multiplication operation.

In more detail, we begin by computing per-blob opacity maps $o \in [0,1]^{H \times W}$. For each grid location $x_{\text{grid}} \in \left\{\left(\frac{w}{W}, \frac{h}{H}\right)\right\}_{w=1,h=1}^{W,H}$ we find the squared Mahalanobis distance to the blob center x:

$$d(x_{\text{grid}}, x) = (x_{\text{grid}} - x)^T (R \Sigma R^T)^{-1} (x_{\text{grid}} - x), \tag{2.1}$$

where $\Sigma = c \begin{bmatrix} a & 0 \\ 0 & \frac{1}{a} \end{bmatrix}$, R is a 2D rotation matrix by angle θ, and $c = 0.02$ controls blob edge sharpness. The opacity of a blob at a given grid location is then:

$$o(x_{\text{grid}}) = \sigma\left(s - d(x_{\text{grid}}, x)\right), \tag{2.2}$$

where s acts as a control of blob size by shifting inputs to the sigmoid. Intuitively, this can be thought of as taking a soft thresholding operation on a Gaussian to define an in-region and an out-region. For example, our model can output a large negative $s < 0$ to effectively "turn off" a blob. Rather than taking the softmax to normalize values at each location, we use the alpha-compositing operation [131], which allows us to model occlusions and object relationships more naturally by imposing a 2.1-D z-ordering [117]:

$$\alpha_i(x_{\text{grid}}) = o_i(x_{\text{grid}}) \prod_{j=i+1}^{k} (1 - o_j(x_{\text{grid}})). \tag{2.3}$$

Lastly, our blob mapping network also outputs background vectors ϕ_0, ψ_0, with a fixed opacity $o_0 = 1$. Final features at each grid location are the convex combination of blob feature vectors, given by the (k+1) α_i scores.

2.3.2 From blob layouts to scene images

We now describe a function G that converts the representation of scenes as blobs $\boldsymbol{\beta}$ described in Section 2.3.1 into realistic, harmonized images. To do so, we build on the architecture of StyleGAN2 [74]. We modify it to take in a spatially-varying input tensor based on blob structure features rather than a single, global vector, and perform spatially-varying style modulation.

As opposed to standard StyleGAN, where the single style vector w must capture information about all aspects of the scene, our representation separates layout (blob locations and sizes) and appearance (per-blob feature vectors) by construction, naturally providing a foundation for a disentangled representation.

Concretely, we compute a feature grid Φ at 16×16 resolution using blob structure vectors ϕ_i and use Φ as input to G, removing the first two convolutional blocks of the base architecture to accommodate the increased resolution. We also apply spatial style-based modulation [123] at each convolution using feature grids $\Psi_{l \times l}$ for $l \in \{16, 32, \ldots, 256\}$ computed from blob style vectors ψ_i.

2.3.3 Encouraging disentanglement

Intuitively, all activations within a blob are governed by the same feature vector, encouraging blobs to yield image regions of self-similar properties, *i.e.* entities in a scene. Further, due to the locality of convolution, the layout of blobs in the input must strongly inform the final arrangement of image regions. Finally, our latent space separates layout (blob location, shape, and size) from appearance (blob features) by construction. All the above help our model learn to bind individual blobs to different objects and arrange these blobs into a coherent layout, disentangling scenes spatially into their component parts.

To further nudge our network in this direction, we stochastically perturb blob representations $\boldsymbol{\beta}$ before inputting them to G, enforcing our model to be robust under these perturbations. We implement this by corrupting blob parameters with uniform noise δx, δs, and $\delta\theta$. This requires that blobs be independent of each other, promoting object discovery and discouraging degenerate solutions which rely on precise blob placement or shape.

We also experiment with style mixing, where with probability 0.2 we uniformly sample between 0 and k blobs to swap, and permute style vectors for these blobs among different batch samples. We find that this intervention harms our training process since it requires that all styles match all layouts, an assumption we show does not hold in Section 2.4.3. We also try randomly removing blobs from the forward process with some probability, but found this hurts training, since certain objects must always be present in certain kinds of scenes (*e.g.* kitchens are unlikely to have no refrigerator). This constraint led to a more distributed, and therefore less controllable, representation of scene entities.

2.4 Experiments

We evaluate our learned representation quantitatively and qualitatively and demonstrate that a spatially disentangled representation of scenes emerges. We begin by showing that our model learns to associate individual blobs with objects in scenes, and then show that our representation captures the distribution of scene layouts. We highlight some applications of our model in Figure 2.3. Finally, we use our model to parse the layouts of real scene images via inversion. For more results, including on additional datasets and ablations, please see Section 2.6.

2.4.1 Training and implementation

We largely follow the training procedure set forth in StyleGAN2 [74], with nonsaturating loss [41], R1 regularization every 16 steps with $\gamma = 100$ but no path length regularization, and exponential moving average of model weights [76]. We use the Adam optimizer [79] with learning rate 0.002 and implement equalized learning rate for stability purposes as recommended by [74, 76].

We set $d_{\text{noise}} = 512$. Our layout generator F is an 8-layer MLP with $d_{\text{hidden}} = 1024$ and leaky ReLU activations. We L2-normalize ϕ and ψ vectors output by the layout generator

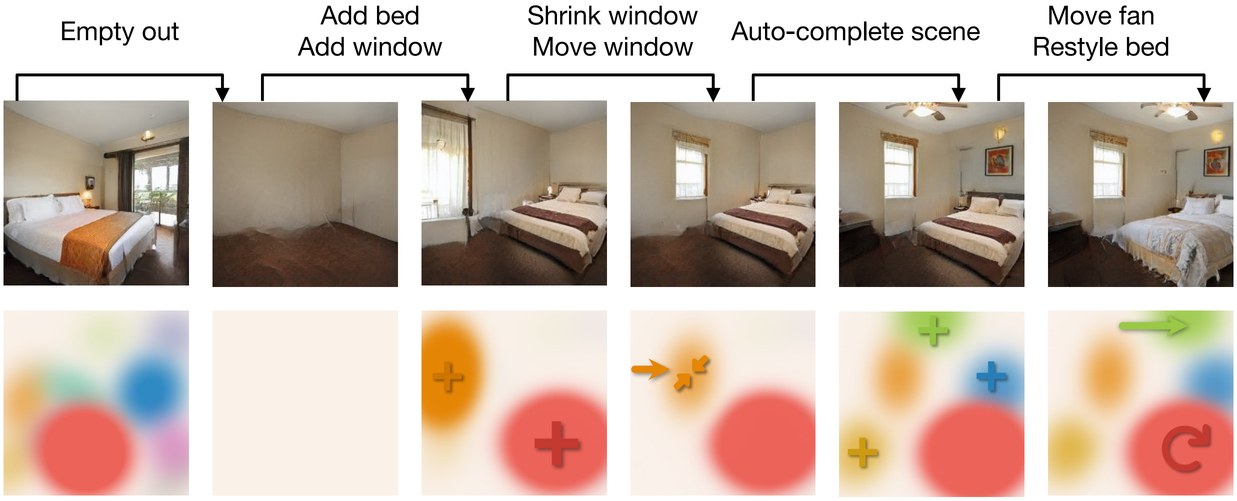


Figure 2.3: **Blobs allow extensive image manipulation:** We apply a sequence of modifications to the blob map of an image generated by our model and show the resulting images outputs at each stage of the editing process, demonstrating the strength of our learned representation.

before splatting. Altogether, the dimension of the last layer is $d_{\text{out}} = k(d_{\text{in}} + d_{\text{style}} + 5) + d_{\text{in}} + d_{\text{style}}$. To compensate for the removal of the first two convolutional blocks in the generator G, we increase channel widths at all remaining layers by 50%. We set d_{in} and d_{style} depending on the number of blobs k, and values range between 256 and 768. We experiment with $k \in [5, 50]$ depending on the data considered. We set the blob sigmoid temperature $c = 0.02$ by visual inspection of blob edge hardness. Model performance is relatively insensitive to jittering parameters. We perturb blob parameters with uniform noise as $\delta x \in [-0.04, 0.04]$ (around 10px at 256px resolution), $\delta\theta \in [-0.1, 0.1]$ rad (around 6)$, and \delta s \in [-0.5, 0.5]$ (varying radii of blobs by around 5px).

We train our model on categories from the LSUN scenes dataset [194]. In particular, we train models on bedrooms; conference rooms; and the union of kitchens, living rooms, and dining rooms. In the following section, we show results of models trained on bedroom data with $k = 10$ blobs. Please see Appendix for results on more data (2.6.1), further details on our blob parametrization and its implementation (2.6.4), and ablations (2.6.7).

2.4.2 Discovering entities with blobs

The ideal representation is able to disentangle complex images of scenes into the objects that comprise them. Here, we demonstrate through various image manipulation applications that this ability emerges in our model. Our unsupervised representation allows effortless

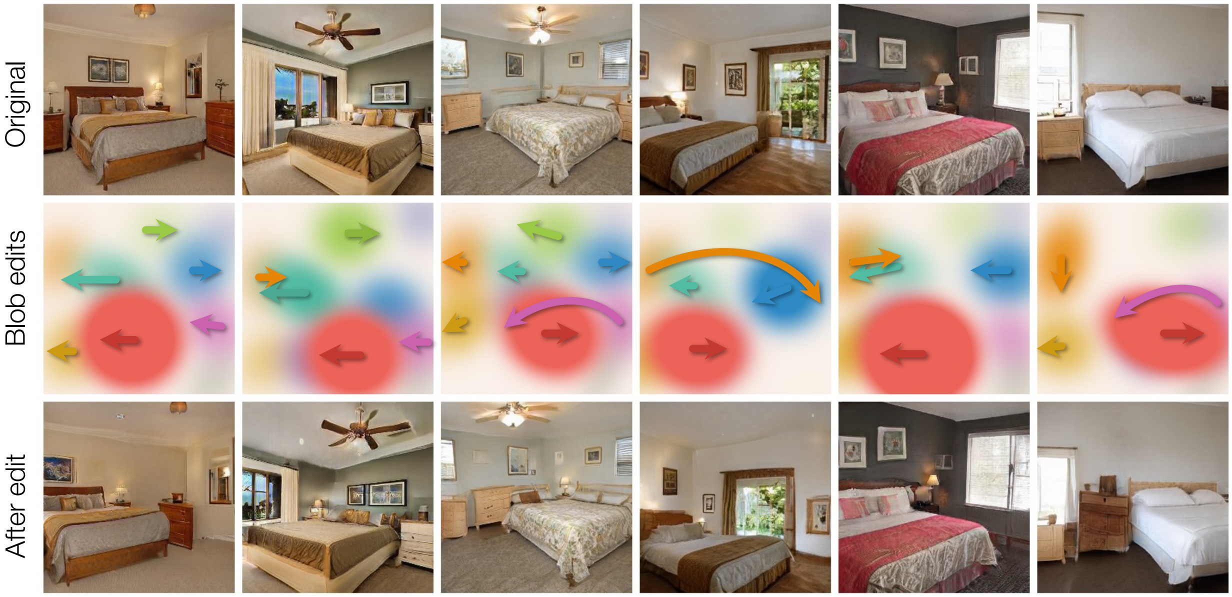


Figure 2.4: **Moving blobs to rearrange objects:** By modifying coordinates x_i of certain blobs as shown in the middle row, we can perform operations such as rearranging furniture in bedrooms. Note that since our representation is layered, we can model occlusions, such as the bed and the dresser in the leftmost and rightmost images.

rearrangement, removal, cloning, and restyling of objects in scenes. We also measure correlation between blob presence and semantic categories as predicted by an off-the-shelf network and thus empirically verify the associations discovered by our model.

Figure 2.4 shows the result of intervening to manipulate the center coordinates x_i of blobs output by our model, and thus **rearranging furniture configurations**. We are able to arbitrarily alter the position of objects in the scene by shifting their corresponding blobs without affecting their appearance. This interaction is related to traditional "image reshuffling" where rearrangement of image content is done in pixel space [162, 6, 145]. Our model's notion of depth ordering also allows us to easily de-occlude objects – *e.g.* curtains, dressers, or nightstands – that were hidden in the original images, while also enabling the introduction of new occlusions by moving one blob behind another.

In Figure 2.5, we show the effect of **removing entirely certain blobs** from the representation. Specifically, we remove all blobs but the one responsible for beds, and show that our model is able to clear out the room accordingly. We also remove the bed blob but leave the rest of the room intact, showing a remarkable ability to create bedless bedrooms, despite training on a dataset of rooms with beds. Figure 2.3 shows the effect of resizing blobs to change window size; see 2.6.1 for further results on changing blob size and shape. In Fig. 2.6, we remove a blob that our model – trained on a challenging multi-category union of scene datasets – has learned to associate with tables across scene categories.

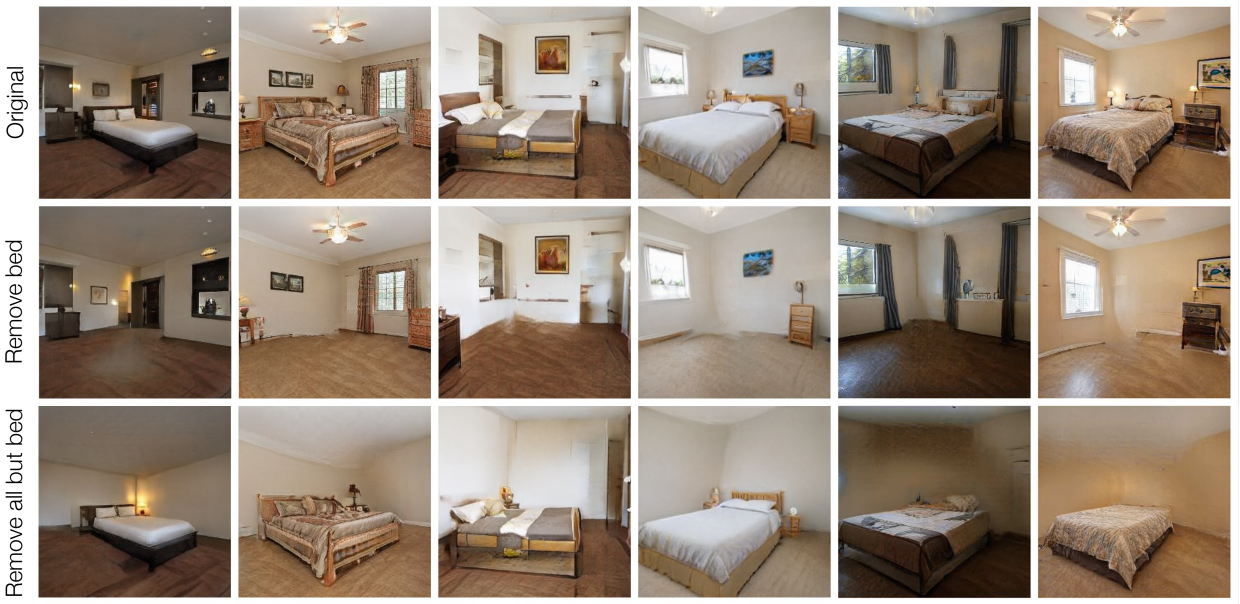


Figure 2.5: **Removing blobs:** Despite the extreme rarity of bedless bedrooms in the training data, the ability to remove beds from scenes by removing corresponding blobs emerges. We can also remove windows, lamps and fans, paintings, dressers, and nightstands in the same manner.

Our edits are not constrained to the set of blobs present in a layout generated by our model; we can also introduce new blobs. Figure 2.8 demonstrates the impact of **copying and pasting the same blob** in a new location. Our model is able to faithfully duplicate objects in scenes even when the duplication yields an image that is out of distribution, such as a room with two ceiling fans.

Our representation also allows performing edits across images. Figure 2.7 shows the **highly granular redecorating** enabled by swapping blob style vectors; we are able to copy objects such as bedsheets, windows, and artwork from one room to another without otherwise affecting the rendered scene.

Quantitative blob analysis. Next, we quantitatively study the strong associations between blobs and semantic object classes. We do so by randomly setting the size parameter s of a blob to a large negative number to effectively remove it. We then use an off-the-shelf segmentation model to measure which semantic class has disappeared. We visualize the correlation between classes and blobs in Figure 2.9 *(left)*; the sparsity of this matrix shows that blobs learn to specialize into distinct scene entities. We also visualize the distribution of blob centroids in Figure 2.9 *(right)*, computed by sampling many different random vectors z. The resultant heatmaps provide a glimpse into the distribution of objects in training data – our model learns to locate blobs at specific image regions and control the objects they represent by varying feature vectors.

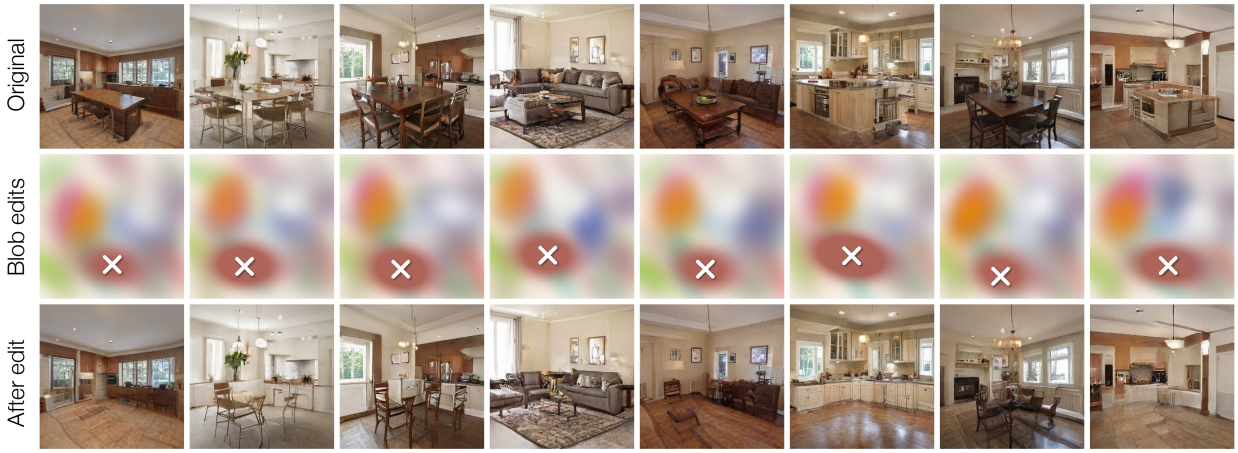


Figure 2.6: **Removing all sorts of tables:** We train BlobGAN on a multi-category dataset of kitchens, living rooms, and dining rooms. We find that a particular blob specializes in generating tables across scene types, and feature vectors dictate whether it becomes a coffee table, kitchen island, or dining table. For many more editing operations on this dataset and others, please see Appendix.

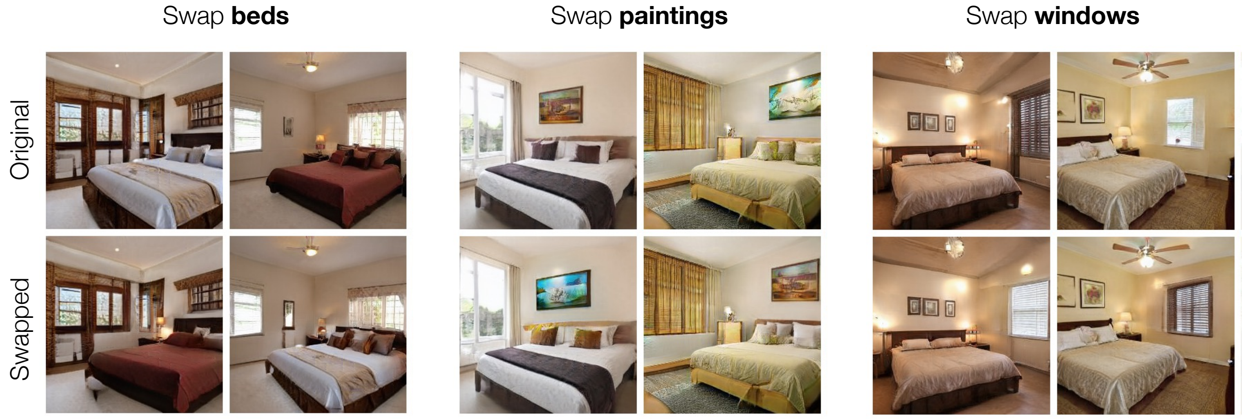


Figure 2.7: **Swapping blob styles:** Interchanging ψ_i vectors without modifying layout leads to localized edits which change the appearance of individual objects in the scene.

2.4.3 Composing blobs into layouts

The ideal representation of scenes must go beyond simply disentangling images into their component parts, and capture the rich contextual relationships between these parts that dictate the process of scene formation [14, 57]. In contrast to previous work in generative modeling of realistic images, our representation explicitly discovers the layout (*i.e.*, the joint distribution) of objects in scenes.

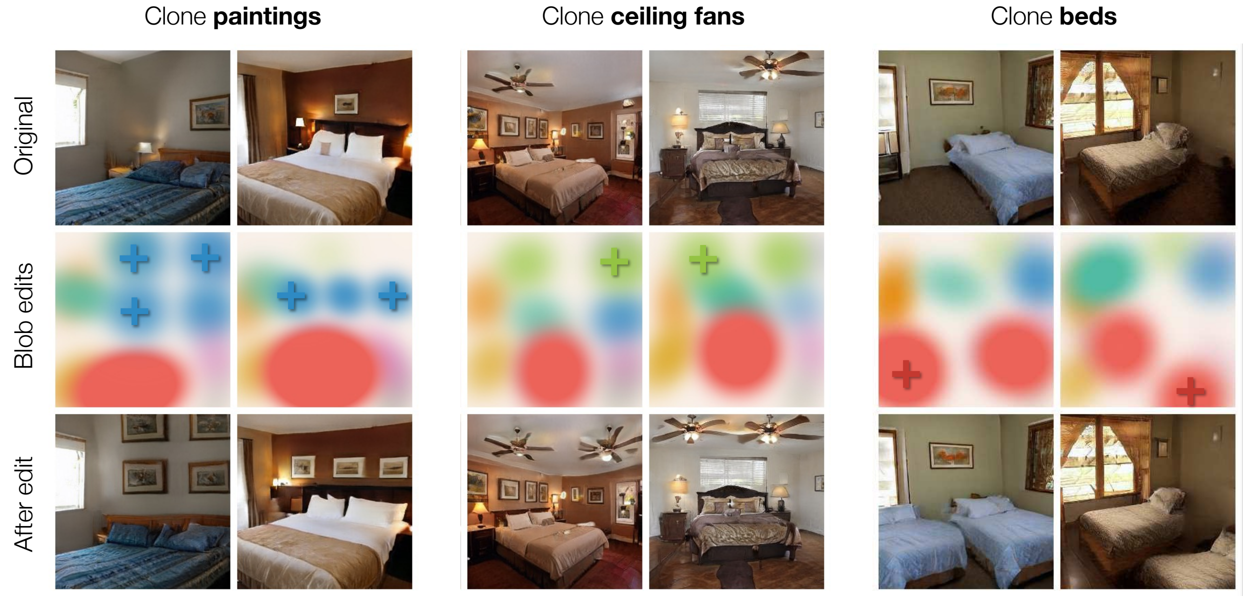


Figure 2.8: **Cloning blobs:** We clone blobs in scenes, arrange them to form a new layout, and show corresponding model outputs. Added blobs are marked with a +.

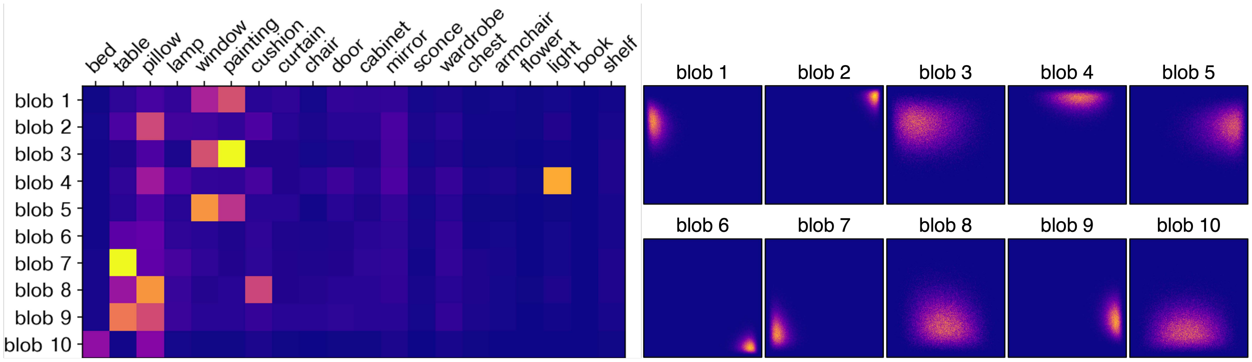


Figure 2.9: **Blob spatial preferences:** Our model allocates each blob to a certain region of the image canvas, revealing patterns in the training distribution of objects. We visualize each blob's correlation with classes predicted by a segmentation model [168] *(left)* as well as the spatial distribution of blob centroids *(right)*.

By solving a simple constrained optimization problem at test-time, we are able to sample realistic images that satisfy constraints about the underlying scene, a functionality we call "scene auto-complete". This auto-complete allows applications such as filling empty rooms with items, plausibly populating rooms given a bed or window at a certain location, and finding layouts that are compatible with certain sets of furniture.

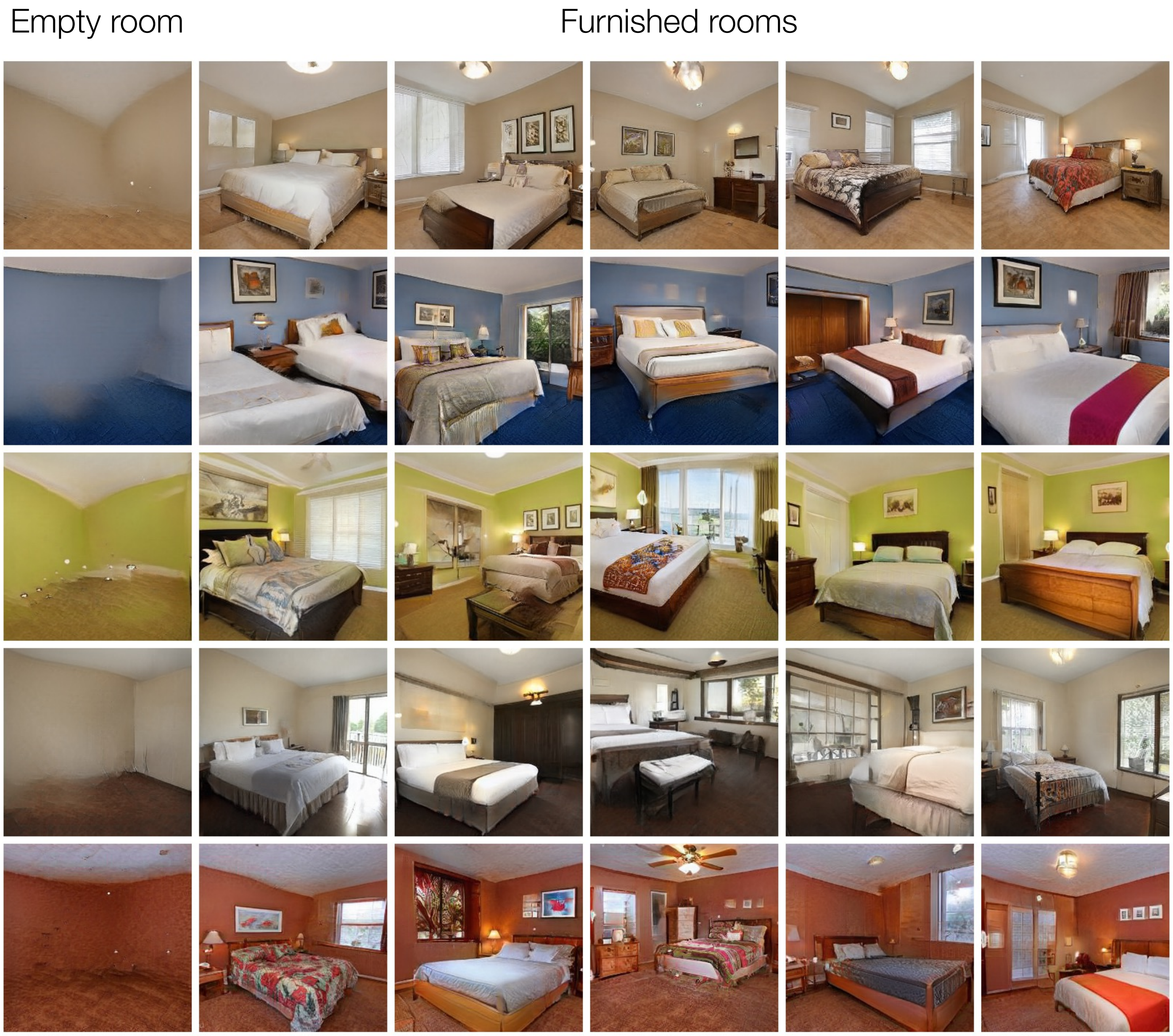


Figure 2.10: **Generating and populating empty rooms:** We show different empty rooms, each with their own background vector ψ_0, as well as furnished rooms given by latents z optimized to match these background vectors. This simple sampling procedure yields a diverse range of layouts to fill the scenes. Note that while empty rooms do not appear in training data, our model is reasonably capable of generating them.

We ground this ability quantitatively by demonstrating that "not everything goes with everything" [18] in real-world scenes – for example, not every room's style can be combined with any room's layout. We show that our scene auto-complete yields images that are significantly more photorealistic than naïvely combining scene properties at random, and outperforms regular StyleGAN in image quality, diversity as well as in fidelity of edits.

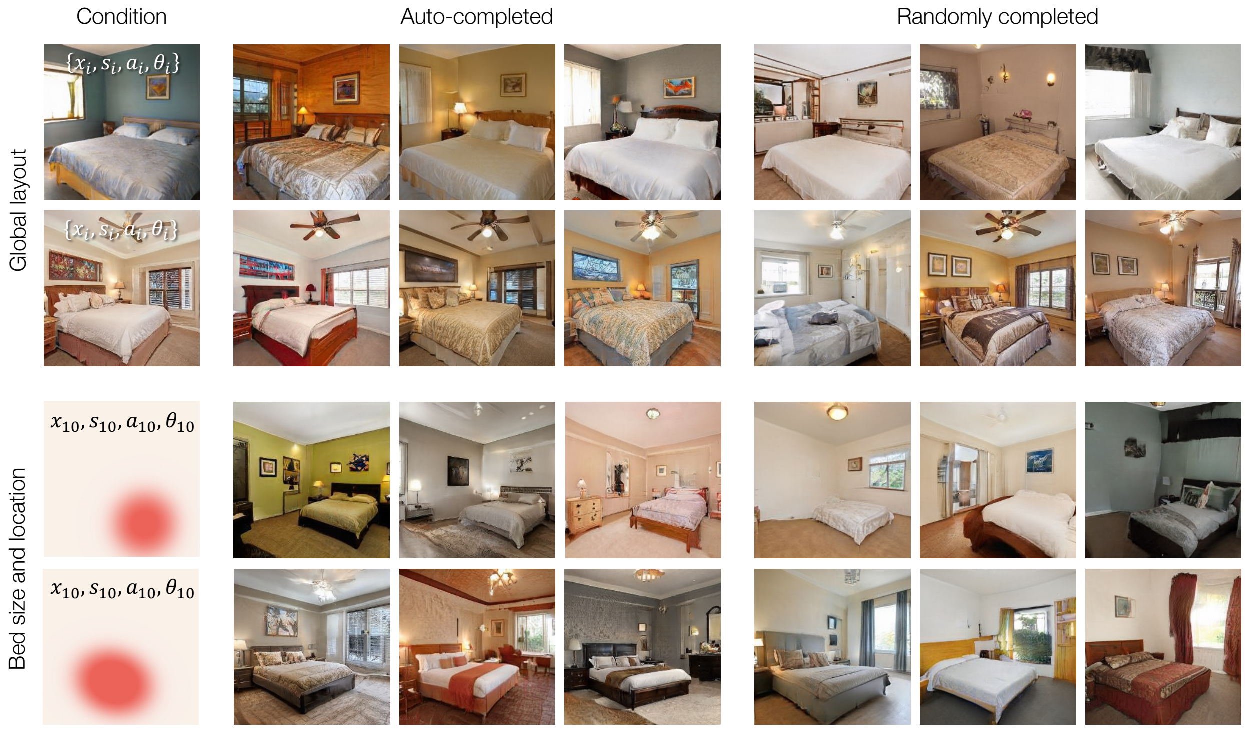

Condition
Auto-completed
Randomly completed
Global layout
Bed size and location
$\{x_i, s_i, a_i, \theta_i\}$
$\{x_i, s_i, a_i, \theta_i\}$
$x_{10}, s_{10}, a_{10}, \theta_{10}$
$x_{10}, s_{10}, a_{10}, \theta_{10}$

Figure 2.11: **Scene auto-complete:** Various conditional generation problems fall under the umbrella of "scene auto-complete", *i.e.* using our layout network F to sample different scenes satisfying constraints on a subset of blob parameters. We show layout-conditioned style generation as well as prediction of plausible scenes given the location and size (but not style) of beds. Rather than using F to plausibly auto-complete scenes, we can also generate a random scene and simply override parameters of interest to match desired values. As shown on the right, such scenes have objects inserted, removed, reoriented, or otherwise disfigured due to incompatibility.

Conditionally sampling scenes: We can construct an ad-hoc conditional distribution by optimizing random inputs to match a set of constraints in the form of properties c of a source image's blob map $\boldsymbol{\beta}$:

$$c \subset \bigcup_{i=0}^{k} \{x_i^{\text{src}}, s_i^{\text{src}}, a_i^{\text{src}}, \theta_i^{\text{src}}, \phi_i^{\text{src}}, \psi_i^{\text{src}}\} \tag{2.4}$$

For example, $c = \{\phi_0^{\text{src}}, \psi_0^{\text{src}}\}$ constrains the background of an output image to match that of a source image, and $c = \{x_i^{\text{src}}, s_i^{\text{src}}, a_i^{\text{src}}\}$ constrains the shape (but not the appearance) of the i-th blob to match the source.

We obtain conditional samples by drawing initial noise vectors $z^{\text{init}} \sim \mathcal{N}(0, I_d)$ and optimizing $F(z^{\text{init}})$ to match the constraint set c with an L2 loss, leaving other parameters

		Layout → Styles				Window → Room				Bed → Room				Painting → Room			
		FID ↓	PD ↑	GD ↑	C ↑	FID	PD	GD	C	FID	PD	GD	C	FID	PD	GD	C
StyleGAN	2 coarse	4.23	0.75	0.77	46.5	-	-	-	-	-	-	-	-	-	-	-	-
	3 coarse	5.04	0.73	0.76	55.3	-	-	-	-	-	-	-	-	-	-	-	-
	4 coarse	5.58	0.71	0.76	62.9	-	-	-	-	-	-	-	-	-	-	-	-
BlobGAN	Random	8.10	0.72	0.74	47.9	6.41	0.72	0.73	17.4	10.88	0.67	0.73	52.6	6.31	0.72	0.73	7.7
	Conditional	4.59	0.70	0.74	55.2	4.75	0.67	0.72	27.2	7.12	0.64	0.72	60.0	4.58	0.69	0.73	13.0
	+ source Φ	5.06	0.68	0.74	63.6	-	-	-	-	-	-	-	-	-	-	-	-

Table 2.1: **Not everything goes with everything:** We edit images by overriding properties in target images either generated at random or conditionally sampled using our model. By varying the network depth at which we begin to swap styles in StyleGAN, we tune a knob between image quality and edit consistency. To further preserve global layout and improve consistency, our model can also use structure grids Φ from the source image. **PD** = paired distance, **GD** = global diversity, **C** = consistency. In all cases, scene auto-complete outperforms baselines. Metrics are defined in the main text.

free. We use the Adam optimizer with learning rate 0.01 and find that between 50 and 300 iterations, which complete in around a second on an NVIDIA RTX 3090, give z^{optim} vectors that sufficiently match constraints. We then set the final layout to be $c \cup \{\boldsymbol{\beta}^{\text{optim}} \setminus c\}$, *i.e.* the initial constraints combined with the free parameters given by the optimized noise vectors, and decode layouts into images as described in Section 2.3.2. In effect, this process finds new scenes known by our model to be compatible with the specified constraints, as opposed to randomly drawn from an unconditional distribution.

We examine applications of our scene auto-complete and compare it to scenes generated by baseline approaches in Figures 2.10 and 2.11. Scene auto-complete yields images that are both more realistic and more faithful to the desired image operations. We quantitatively demonstrate this in Table 2.1, where we show that using auto-complete to find target images whose properties to apply for conducting edits significantly outperforms the use of randomly sampled targets and/or models such as StyleGAN not trained with compositionality in mind.

To evaluate image photorealism after an edit, we calculate FID [53] on automatically edited images. We must also ensure that image quality does not come at the expense of sample diversity; to this end, we measure the average LPIPS [204] distance between images before and after the edit and refer to this as Paired Distance (**PD**). We also measure the expected distance between pairs of edited images to gauge whether edits cause perceptual mode collapse, and call this Global Diversity (**GD**). Finally, we confirm that our editing operations stay faithful to the conditioning

	FID ↓	Precision ↑	Recall ↑
StyleGAN2	3.85	0.5932	0.4492
BlobGAN	3.43	0.5974	0.4463

Table 2.2: **Evaluating BlobGAN:** BlobGAN achieves visual quality competitive with StyleGAN2 [75] on LSUN Bedrooms. Our samples are more realistic but capture less of the data distribution [84], perhaps by rejecting unconventional or malformed scenes in the training data.

provided. For predicting style from layout,
we simply report the fraction of image pixels
whose predicted class label as output by a segmentation network [168] remains the same. For localized object edits, we report the intersection-over-union of the set of pixels whose prediction was the target class before and after edit. We refer to this metric as Consistency (**C**).

Our results verify the intuition that, *e.g.*, not every configuration of furniture can fit a bed at a given location. Please see 2.6.6 for more results.

2.4.4 Evaluating visual quality and diversity

Our model achieves perceptual realism competitive with previous work. In Table 2.2, we report FID [53] as well as improved precision and recall [84], which capture realism and diversity of samples. Bedroom images generated by our model appear more realistic than StyleGAN's [72], but less diverse. We hypothesize this is due to the design of our representation, which rejects strange scene configurations that cannot be modeled by blobs. When trained on the challenging union of multiple LSUN indoor scene categories, *BlobGAN outperforms StyleGAN2*, indicating an ability to scale to harder data. See 2.6.1 for details.

2.4.5 Parsing images into regions

Though our representation is learned on generated (*i.e.* fake) images, in Figure 2.12 we show that it can represent real images via inversion. We follow best practices [207, 1, 139, 175, 11] for inversion: We train an encoder to predict blob parameters, reconstructing both real and fake images, and then optimize encoder predictions to better reconstruct specific inputs. While this method leads to uneditable, off-manifold latents in previous work [141], we find our blob representation to be more robust in this sense and amenable to naïve optimization. Importantly, we find that the same manipulations described above can be readily applied to real images after inversion. See 2.6.2 for more information.

2.5 Discussion

We present BlobGAN, a mid-level representation for generative modeling and parsing of scenes. Taking random noise as input, our model first outputs a set of spatial, depth-ordered blobs, and then splats these blobs onto a feature grid. This feature grid is used as input to a convolutional decoder which outputs images. While conceptually simple, this approach leads to the emergence of a disentangled representation that discovers entities in scenes and their layout. We demonstrate a set of edits enabled by our approach, including rearranging layouts by moving blobs and editing styles of individual objects. By removing or cloning blobs, we are even able to generate empty or densely populated rooms, though none exist

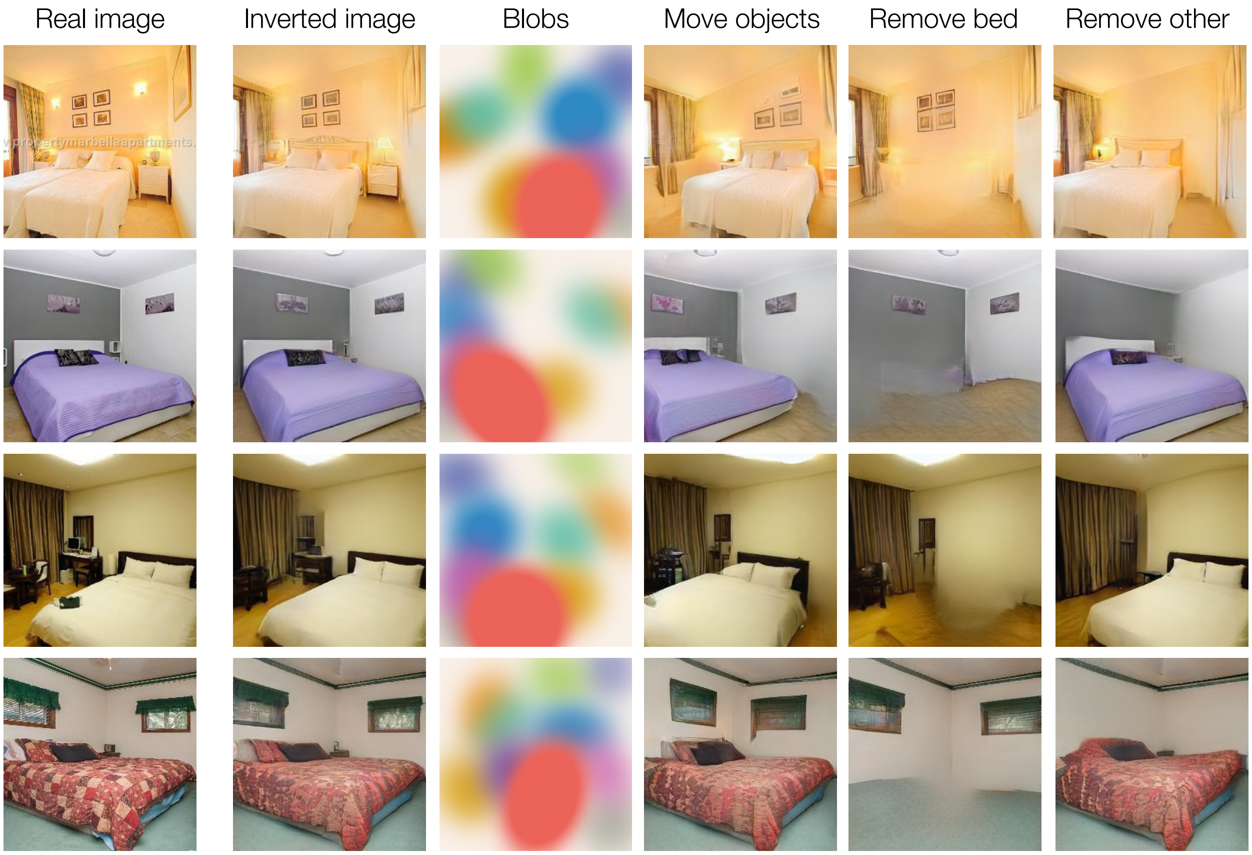


Figure 2.12: **Parsing real images via inversion:** Our representation can also parse real images by inverting them into blob space. We can remove and reposition objects in real images – spot the differences from the original!

in the training set. Our model can also parse and manipulate the layout of real images via inversion.

2.6 Appendix

2.6.1 BlobGAN on other datasets

In the main text, we primarily showed results on LSUN bedrooms [194]. Below, we show that our model can be applied to other datasets and room types. We provide qualitative and quantitative results on our models trained on the challenging LSUN conference room dataset, as well as a joint dataset combining LSUN kitchens, dining rooms, and living rooms [194]. As with bedrooms, our model's images are competitive with previous work in terms of photorealism, and in addition allow extensive manipulation of images. Please see Table 2.3

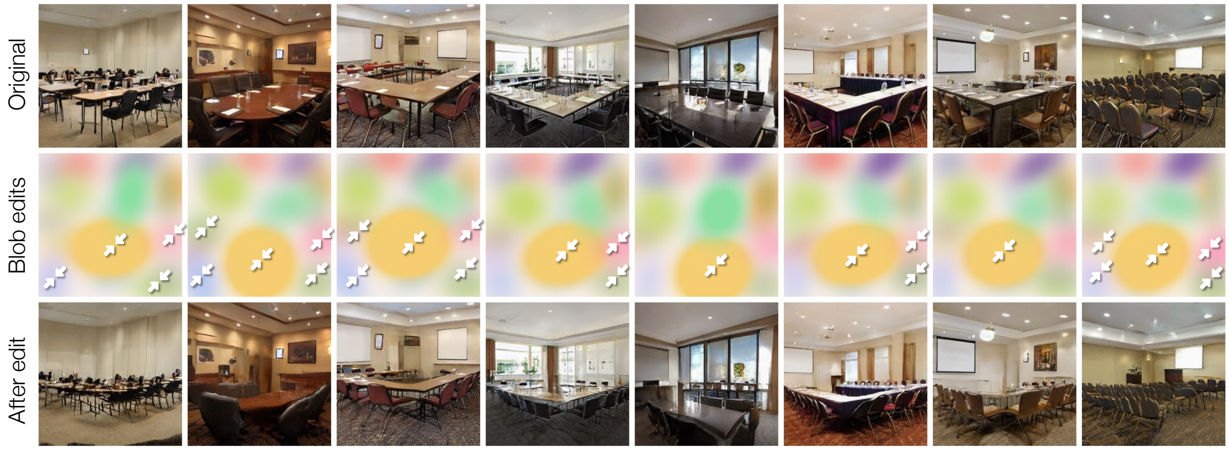


Figure 2.13: **Honey, I shrunk the conference room!** As in Figure 2.3, we show the effect of resizing blobs in generated images. Here, we resize blobs corresponding to tables and chairs, and render identical rooms with shrunken furniture.

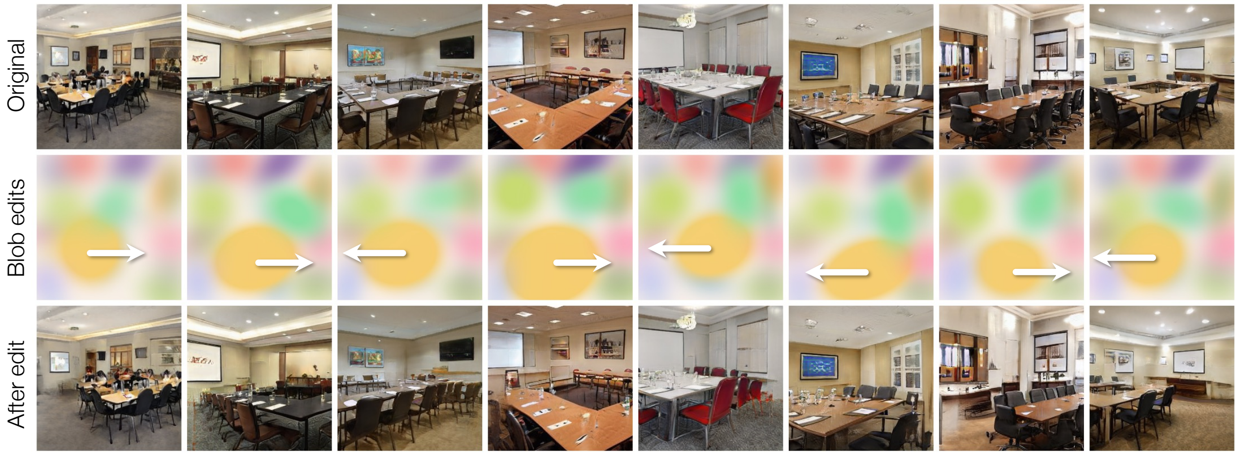


Figure 2.14: **Moving desks and chairs (conference room):** As in Figure 2.4, we show the effect of moving blobs in generated images. Here, we move blobs corresponding to tables and chairs, and render identical rooms with shifted furniture.

for quantitative evaluation. We show image samples and edits on them in Figures 2.19, 2.13, 2.14, 2.15, 2.23, 2.22, and 2.16.

2.6.2 Modeling real images with BlobGAN

We show additional results on inversion and editing of real images in Figures 2.17 and 2.18. Images are drawn from the LSUN bedrooms validation set, which our model does not see

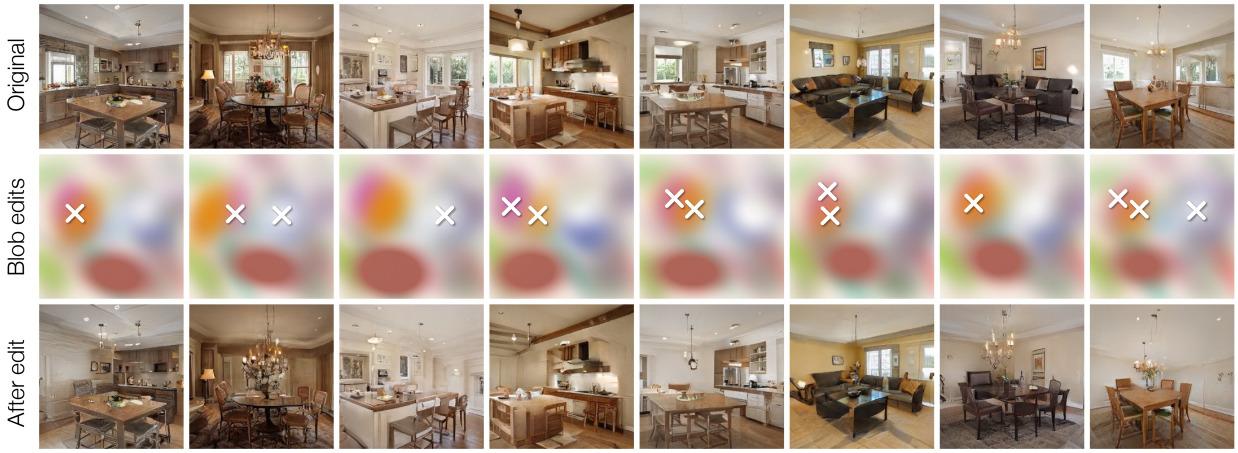


Figure 2.15: **Removing some or all windows (kitchen, living room, dining room):** As shown in Figure 2.5, we can remove windows from complex scenes, though they are often hidden behind cluttered configurations of furniture. We can control which windows to remove by selecting only some of the relevant blobs.

	LSUN Conference			LSUN Kitchen+Living+Dining		
	FID ↓	Precision ↑	Recall ↑	FID ↓	Precision ↑	Recall ↑
StyleGAN2 [75]	6.21	0.5475	0.4554	4.63	0.6005	0.4397
BlobGAN	6.94	0.5297	0.4485	4.41	0.5818	0.4661

Table 2.3: **BlobGAN on diverse data:** On challenging collections of conference rooms and various types of indoor rooms in homes, our model is highly competitive with a StyleGAN2 baseline, while enabling all the applications of the BlobGAN representation. Our model outperforms StyleGAN2 given an equal number of gradient steps (1.5M) on the difficult union of various LSUN indoor scene categories, as measured by FID.

during the training process.

2.6.3 Implementation details

In Section 2.4.5 and Figure 2.12, we demonstrate that real images can be inverted and manipulated with our model. Here, we provide additional details regarding the encoder training procedure. We take an encoder architecture E in the same form as the StyleGAN2 [75] discriminator, without mini-batch statistic discrimination. We use E for inverting images by having the last layer output a long flat vector, which we segment into blob parameters. In addition to reconstructing both real and synthetically generated images with LPIPS [204]

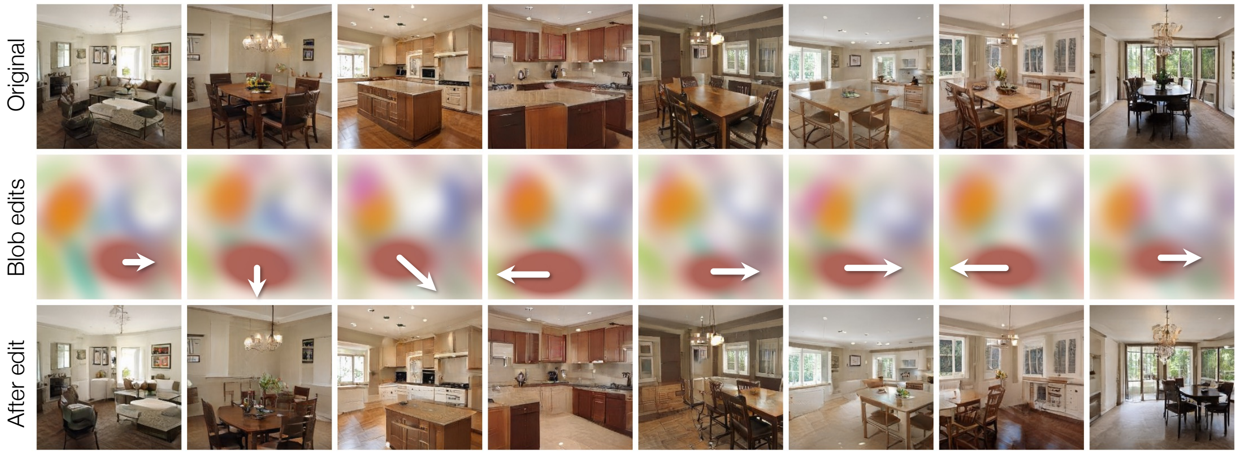


Figure 2.16: **Moving tables and chairs (kitchen, living room, dining room):** Our representation can easily move tables and any associated chairs, by changing the location of blobs 42 (table) and 30 (chairs). Since the two move together, we only show one arrow to represent the edit.

and L2 penalties, we require the parameters $\hat{\boldsymbol{\beta}}$ to match the ground truth parameters $\boldsymbol{\beta}$ in the case of inverting generated images. Our overall loss is:

$$\begin{aligned}\mathcal{L}_{\text{inversion}} = &\mathcal{L}_{\text{LPIPS}}\left(\mathbf{x}_{\text{real}}, G(E(\mathbf{x}_{\text{real}}))\right) + \mathcal{L}_{\text{LPIPS}}(\mathbf{x}_{\text{fake}}, G(E(\mathbf{x}_{\text{fake}}))) \\ &+ \mathcal{L}_2(\mathbf{x}_{\text{real}}, G(E(\mathbf{x}_{\text{real}}))) + \mathcal{L}_2(\mathbf{x}_{\text{fake}}, G(E(\mathbf{x}_{\text{fake}}))) \\ &+ \lambda \mathcal{L}_2(\boldsymbol{\beta}_{\text{fake}}, E(\mathbf{x}_{\text{fake}})),\end{aligned} \tag{2.5}$$

with $\lambda = 10$ controlling the strength of the blob reconstruction loss. Taking the L2 loss on blob parameters as a flattened vector would heavily emphasize reconstructing the high-dimensional features, over the important, low-dimensional, scalar quantities of blob locations and sizes. Instead, we compute L2 separately over each blob attribute and take the mean.

We then further optimize the blob parameters to reconstruct the target image, with LPIPS and L2 losses and the Adam optimizer [79] with learning rate 0.01 for 200 steps. While better fitting the input image, this method potentially deviates from the manifold of latents that yield realistic images [1, 141], thus severely impeding editing abilities. Previously proposed solutions offer regularizations to keep the latents on this “manifold” [187, 205, 188]. However, we find our blob representation to be more robust in this sense, and latents yielded by this naïve optimization still amenable to editing.

2.6.4 Blob parametrization

We represent the blob aspect ratio a as two scalar outputs a_0, a_1, sigmoided and then normalized to have a fixed product $a_0 a_1$; we find this to train more stably than one aspect

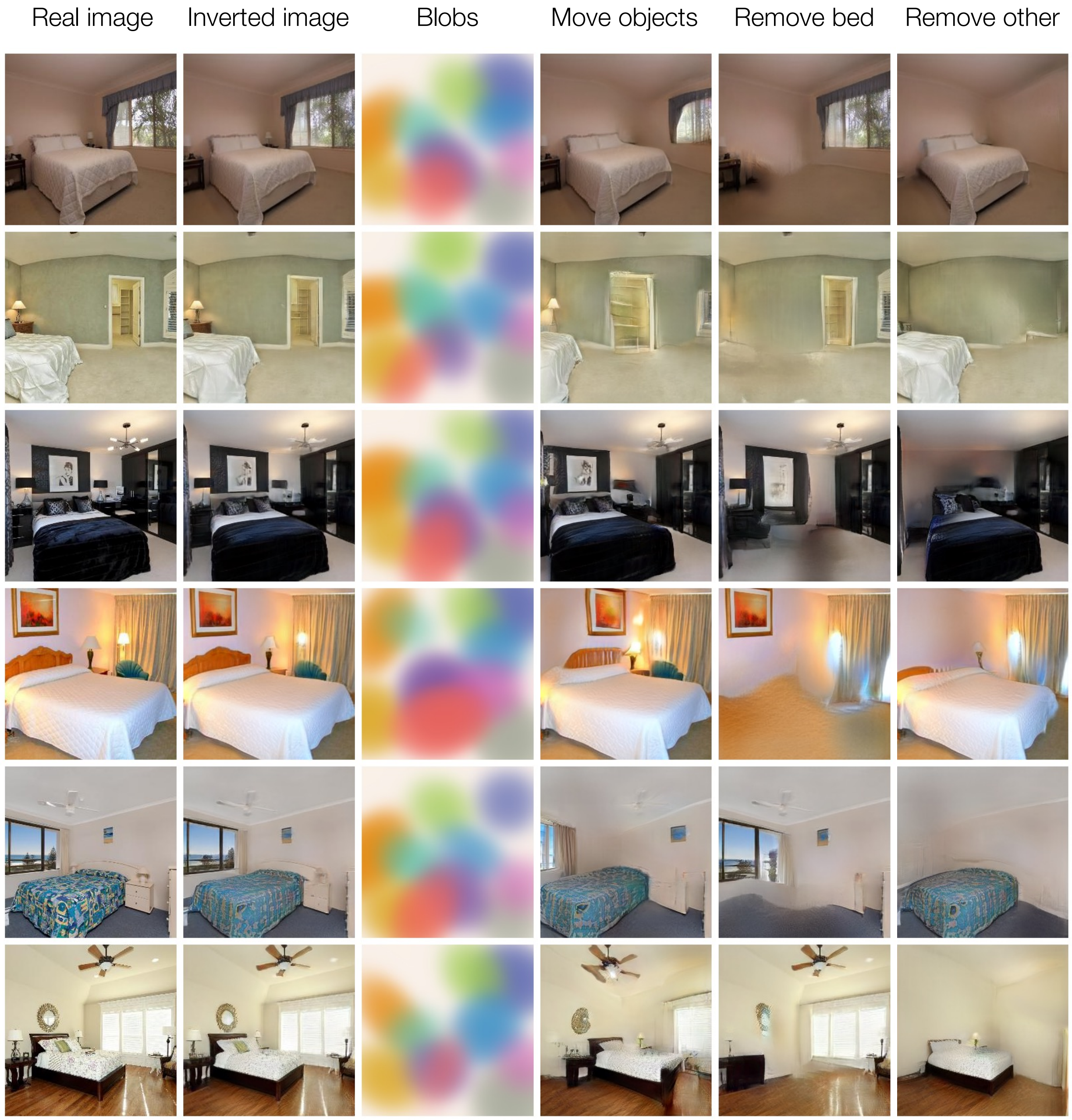


Figure 2.17: **Parsing real images via inversion:** We show the flexibility of our learned representation by applying edits to real images inverted into blob space. We can remove and reposition objects in real images – spot the differences from the original!

ratio. We represent the blob angle θ with two scalars e_0, e_1, from which we construct a unit-normalized axis of rotation $\mathbf{e}$. We find this representation to train far more stably than others, such as regressing to a scalar θ or other parametrizations of Σ like log-Cholesky [102,

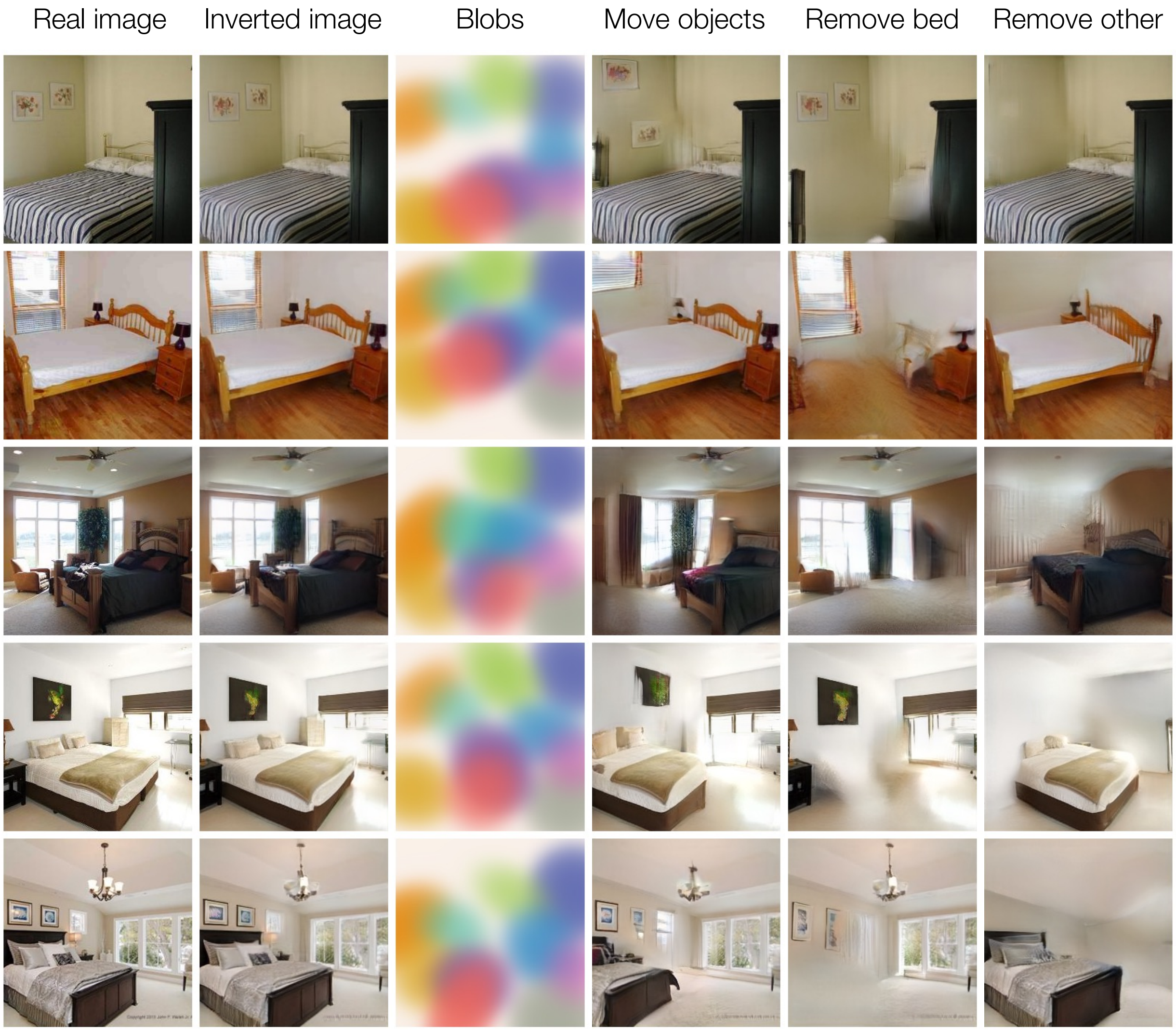


Figure 2.18: **Parsing real images via inversion:** More results on inversion of real images.

155].

We also experimented with alternate representations, such as closed-form ellipses and rectangles as well as Gaussian mixture models. However, we found gradient flow to blob parameters ill-behaved with rectangles and other explicitly defined shapes, even with tricks like gradual opacity falloff, and these models failed to train. With GMMs, depth ordering and occlusions are lost, and blob size and shape depend on other blobs, harming performance. Our model is robust w.r.t. c, and $0.005 \leq c \leq 0.05$ all train well.

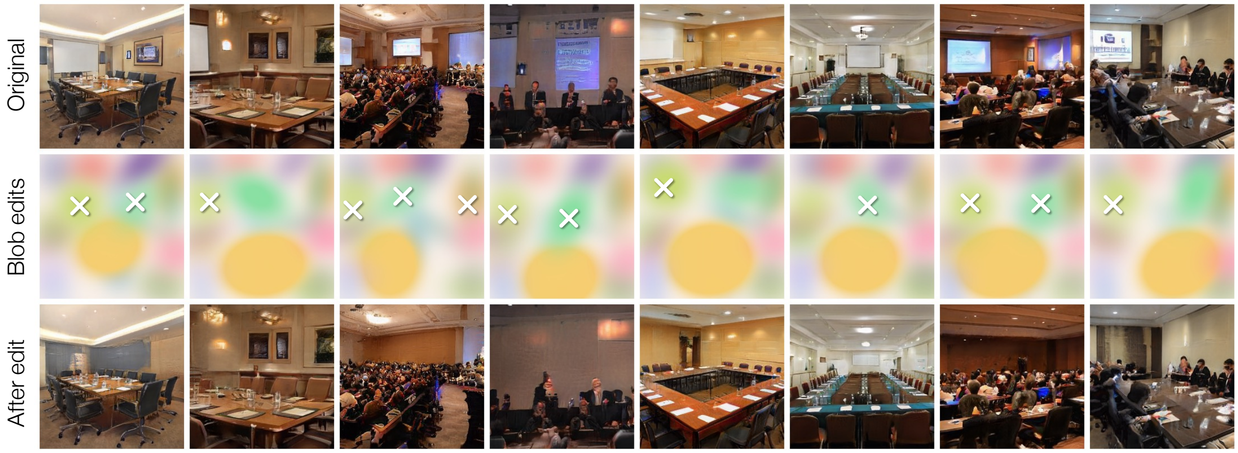


Figure 2.19: **Removing screens in conference rooms.** As in Figure 2.5, we show the effect of removing certain blobs from generated images. Here, we remove blobs corresponding to screens from images of conference rooms.

2.6.5 Limitations

Though our blob representation allows for powerful unsupervised, disentangled scene representations, our model still suffers from various shortcomings. For example, trained networks struggle to disentangle smaller objects (*e.g.* lamps on desks), perspective from object shape, and, occasionally, foreground appearance from background. Further, as shown in the main paper, blobs display a predilection toward certain canvas regions, though whether this is an artifact of dataset bias or model design remains unclear.

2.6.6 Comparison to previous work

In Figures 2.20 and 2.21, we show random samples of untruncated images before and after style swapping. At a given level of photorealism as measured by FID, our model is able to produce layouts far more consistent with the original image thanks to its disentangled, compositional representation.

Lastly, we visualize the trade-off between the precision and recall metric [84] as we change the truncation value in Figure 2.24. Our model generates more perceptually realistic images than StyleGAN at all truncation values $0.0 \leq w \leq 1.0$, although the maximal recall at $w = 1.0$ is lower. In particular, our untruncated model performs better at both precision and recall than *all StyleGAN-generated images with* $w < 0.7$. These results provide evidence for the suggestion that our model's FID is higher because it cannot properly model outlier bedroom scenes using the blob representation.

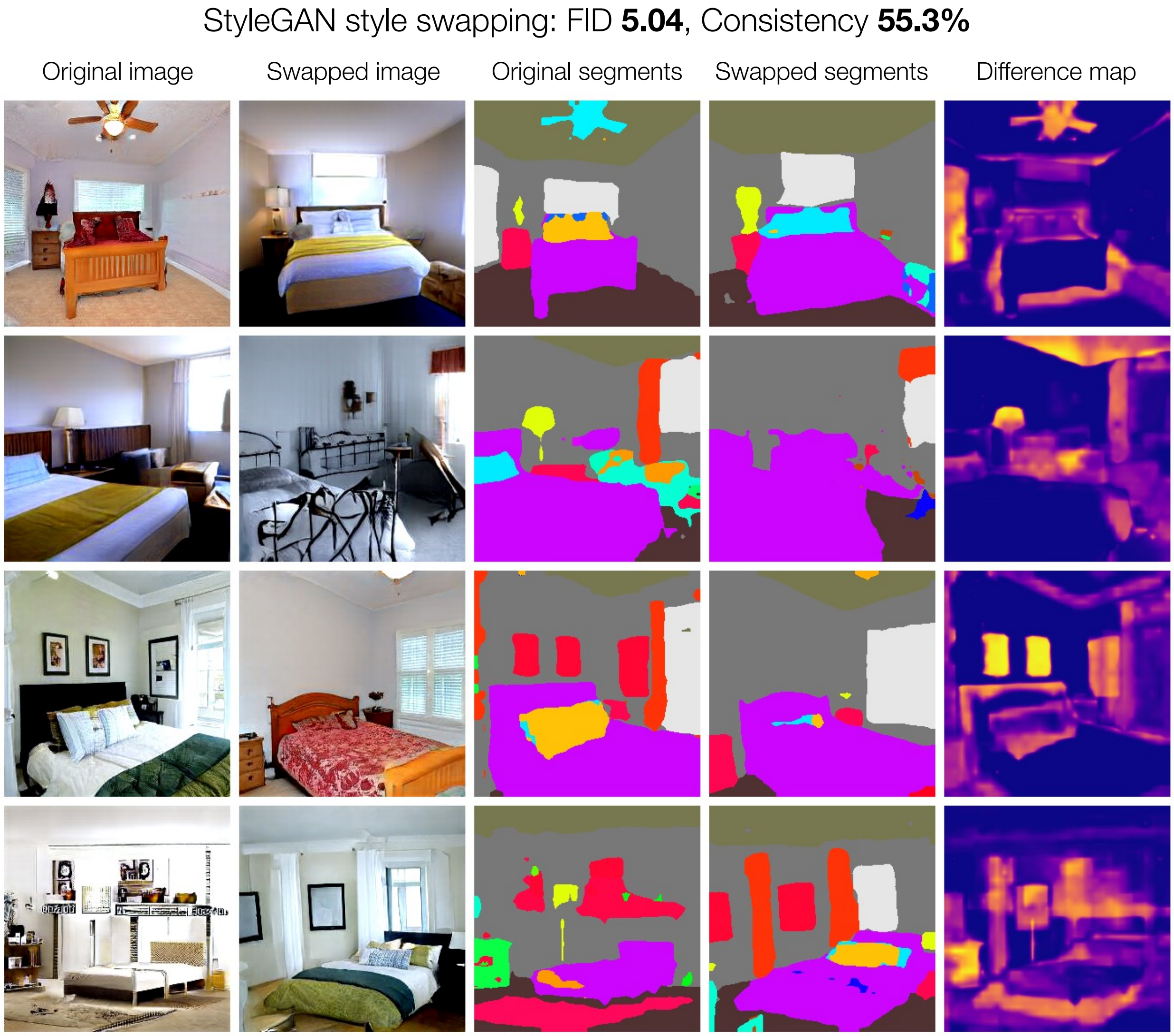


Figure 2.20: **Style swapping with StyleGAN:** We show randomly sampled untruncated StyleGAN images before and after style swapping at layer 4, attaining an FID of 5.04 and layout consistency of 55.3%. The difference map shows the normalized KL divergence of the predicted per-pixel logits before and after swapping.

2.6.7 Model implementation

Hyperparameters and training

For the bedroom model trained in the paper, we use $d_{\text{in}} = 768$ and $d_{\text{style}} = 512$. Our generator with $k = 10$ blobs has 57.2 million parameters: 21.3 million in F and the remaining 35.9 million in G.

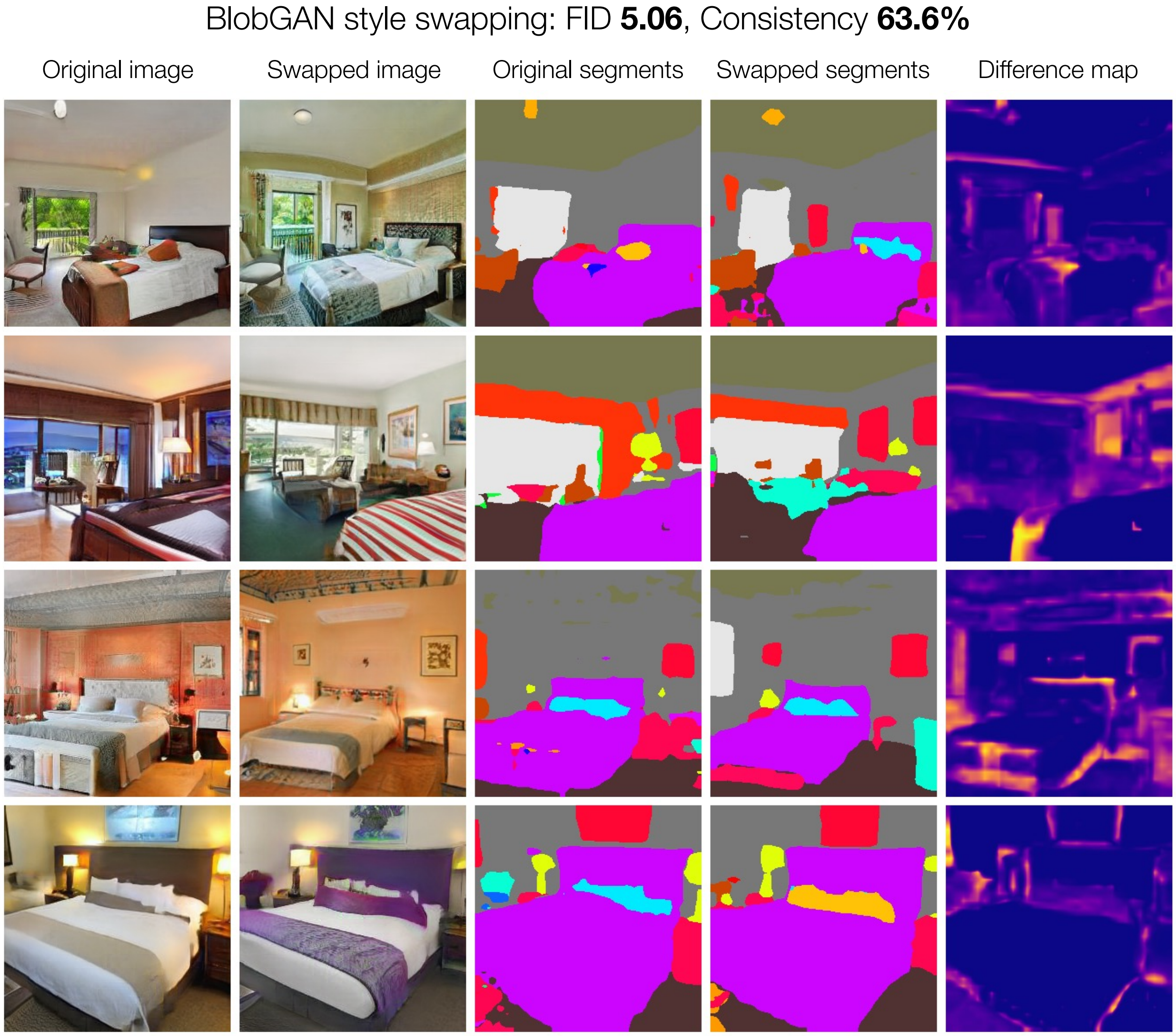


Figure 2.21: **Style swapping with BlobGAN:** We show randomly sampled untruncated BlobGAN images before and after style swapping, attaining an FID of 5.06 and layout consistency of 63.6%. The difference map shows the normalized KL divergence of the predicted per-pixel logits before and after swapping.

The model trained on LSUN conference rooms uses $k = 20$ and has 34.5M parameters in F; all other hyperparameters are as in the bedroom model.

The model trained on the union of LSUN kitchens, living rooms, and dining rooms uses $k = 45$ due to the increased complexity of the combined dataset, and thus reduces $d_{\text{in}} = 256$ and $d_{\text{style}} = 256$. This model has 61.3 million parameters in the generator: 31.3M in F and 30.0M in G.

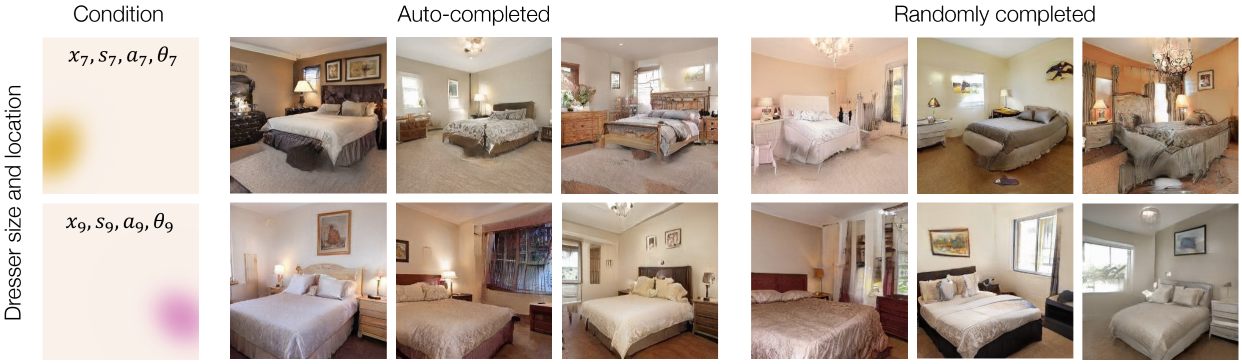

Condition
Auto-completed
Randomly completed
Dresser size and location
x_7, s_7, a_7, θ_7
x_9, s_9, a_9, θ_9

Figure 2.22: **Scene auto-complete:** Various conditional generation problems fall under the umbrella of “scene auto-complete”, *i.e.* optimizing random noise vectors to match a set of blob parameters when run through our layout network F. We show prediction of plausible scenes given the location and size (but not style) of dressers and nightstands. The model must not only predict the arrangement of the missing blobs, but also assign all blobs realistic appearance. When sampling target images randomly, objects are often randomly inserted, removed, reoriented, or otherwise disfigured due to incompatibility.

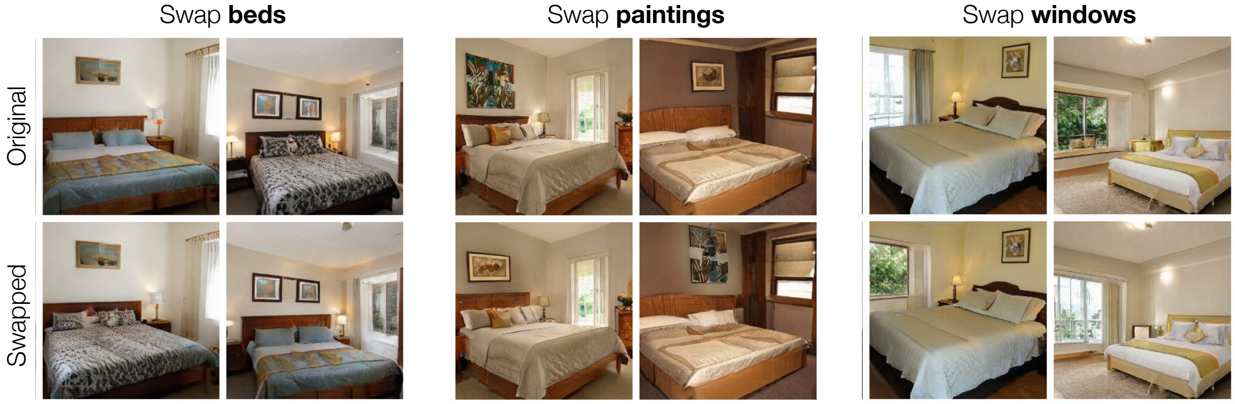


Figure 2.23: **Swapping blob styles:** Interchanging ψ_i vectors without modifying layout leads to localized edits which change the appearance of individual objects in the scene.

We train all models for 1.5 million gradient steps with batch size 24 per-GPU across 8 NVIDIA A100 GPUs, except the bedrooms models (both BlobGAN and StyleGAN2), which are trained for 2.8 million steps. On the bedrooms model, we experiment with $k = 20$ blobs as well as $k = 10$ blobs with no jitter. We find that results are less interpretable with 20 blobs and disentanglement is lower, perhaps since the model can “approximate” slightly higher-frequency data by using more blobs. This model also has a worse FID of 3.73. We also train a model with $k = 10$ blobs and no jitter, which attains comparable FID to the

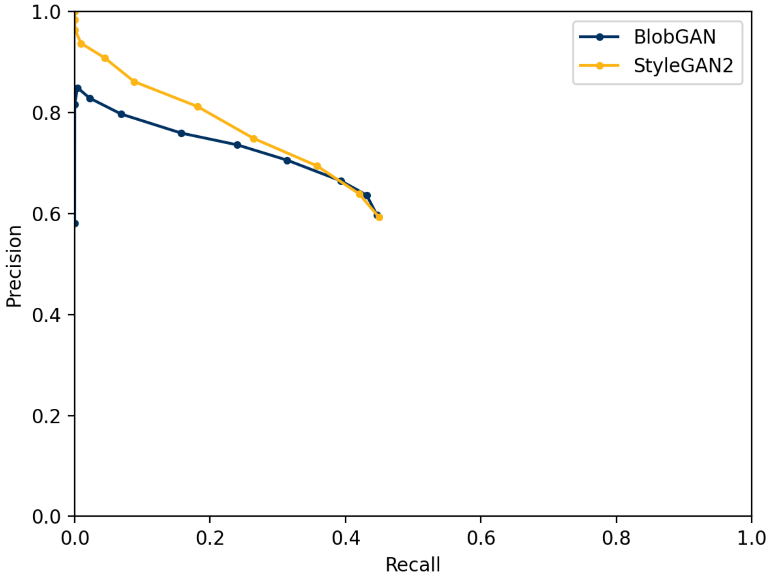


Figure 2.24: We plot the precision-recall curve, by varying truncation values w, on LSUN bedrooms. Our untruncated model outperforms StyleGAN2 [75] with truncation values $w < 0.7$ in both precision and recall of generated images. While still outperforming StyleGAN2 on FID (Table 2.2), our model operates at a different point on this curve than StyleGAN2 – higher precision and lower recall – supporting the hypothesis that BlobGAN's FID suffers due to its inability to model long-tail, oddly-formed scenes.

model with jitter, but with slightly reduced editing capabilities. Across all experiments, we find that changing k minorly impacts FID. Extra blobs mostly go unused, but too few blobs mean objects cannot be properly separated.

Image sampling

We sample all images shown in the paper and Supplementary Material with truncation. We truncate latents at the penultimate layer of F, since truncating in blob parameters space leads to undesirable behavior (*e.g.* biasing blob coordinates toward the center of the image). Then, truncation of random noise vector z's output of blob parameters $\boldsymbol{\beta}$ with a weight of w gives:

$$\boldsymbol{\beta}_{\text{trunc}} = F_L\left((1-w) \mathop{\mathbb{E}}_{z'\sim\mathcal{N}(0,I)}[F_{0:-1}(z')] + wF_{0:L-1}(z)\right) \tag{2.6}$$

Where $F_{l:m}$ represents layers l through m, inclusive, of the network which has L layers total. In practice, we approximate the expectation by sampling 100,000 random noise vectors. We use $w = 0.6$ or $w = 0.7$ for all bedroom images. $w = 0.5$ for images of conference rooms, and $w = 0.4$ for other indoor scenes, except when indicated otherwise ($w = 1$ means no truncation).

Object style swapping

When swapping styles between objects, rather than splatting the target (new) object's style $\psi_{i,\text{tgt}}$ directly onto the source (original) image's background style $\psi_{\text{bg, src}}$, we interpolate first between $\psi_{i,\text{tgt}}$ and then $\psi_{\text{bg,tgt}}$ (*i.e.*, the target image's background) at the border of the blob, and then splat this onto the background $\psi_{\text{bg, src}}$.

We find this necessary since the model learns to treat features on the border of a blob, which are typically a convex combination of the blob feature and the background feature, as belonging to the blob; when an unanticipated background feature becomes part of the feature along the border, the model is more prone to producing artifacts. This simple procedure mitigates this undesirable behavior and is trivially fully automated.

Spatial modulation

In StyleGAN2, convolution weights at layer l, $\theta_l \in \mathbb{R}^{d_l \times d_{l-1} \times k \times k}$, are multiplied by an affine-transformed style vector $w \in \mathbb{R}^d_l$ and then unit-normalized to perform modulation. Since our modulation varies spatially, we instead multiply input feature maps $x_{l-1} \in \mathbb{R}^{d_{l-1} \times h \times w}$ by a unit-normalized style grid $\Psi_l \in \mathbb{R}^{d_\text{style} \times h \times w}$ with a per-pixel affine transform, before convolving with unit-normalized weights θ_l to output new feature maps x_l. Affine transforms f map from d_style to d_l. More specifically, in StyleGAN2, modulated convolution is implemented as:

$$x_l = x_{l-1} * \frac{f(w_l) \odot \theta_l}{\|f(w_l) \odot \theta_l\|_2} \tag{2.7}$$

Since our styles are spatially varying, we cannot multiply convolution weights by the same broadcasted tensor throughout, and must modify our modulation:

$$x_l = \left(x_{l-1} \odot \frac{f(\Psi_l)}{\|f(\Psi_l)\|_2} \right) * \frac{\theta_l}{\|\theta_l\|_2} \tag{2.8}$$

We find this normalization scheme, also used in [123, 124], to work well in practice despite not having the same statistical guarantees as the original derivation.

Uncurated samples

In Figures 2.25 and 2.26, we show randomly sampled images from our model and StyleGAN2 trained on LSUN Bedrooms. We show the same on LSUN kitchens, living rooms, dining rooms, and conference rooms in Figures 2.27, 2.28, 2.29, and 2.30.

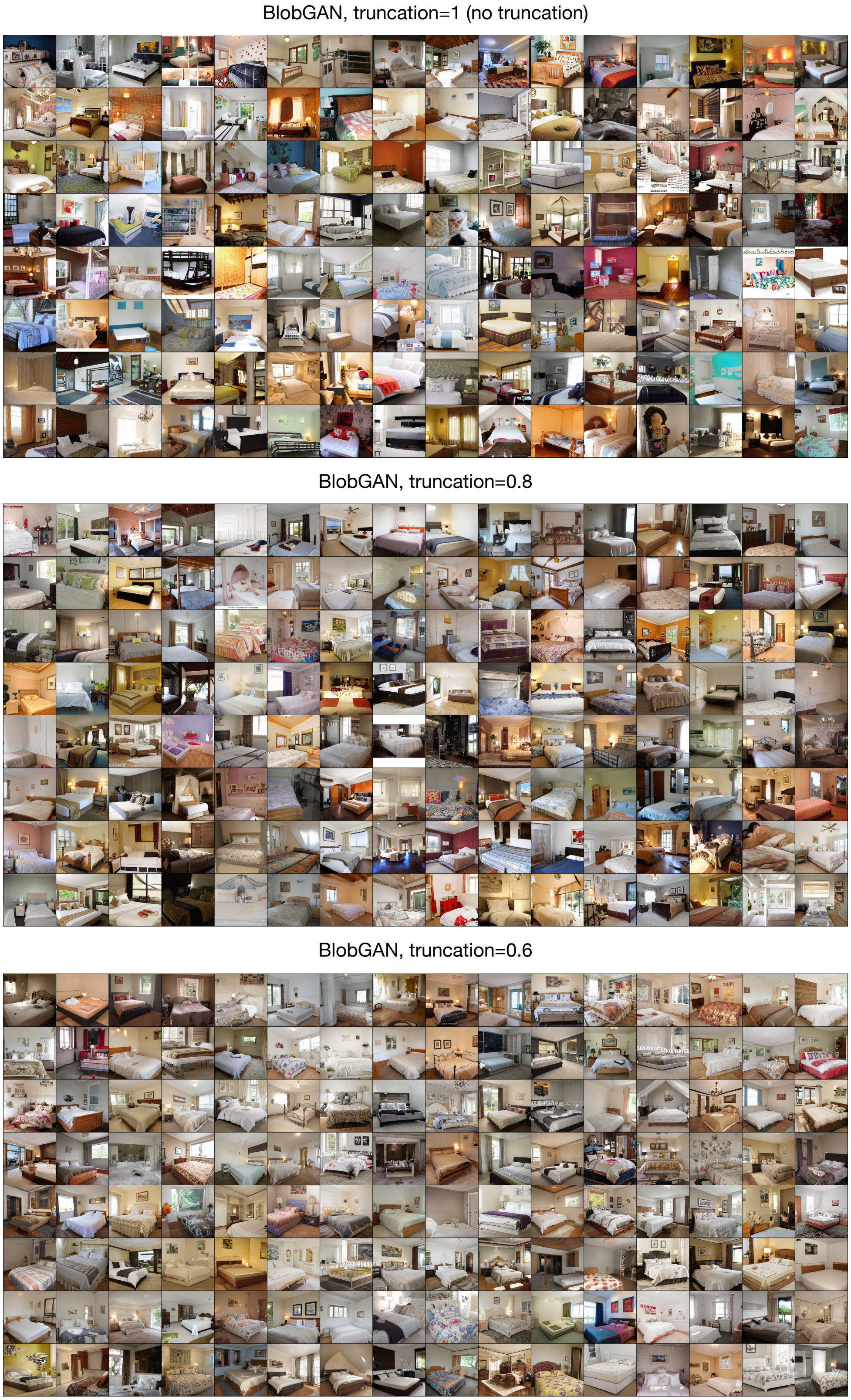
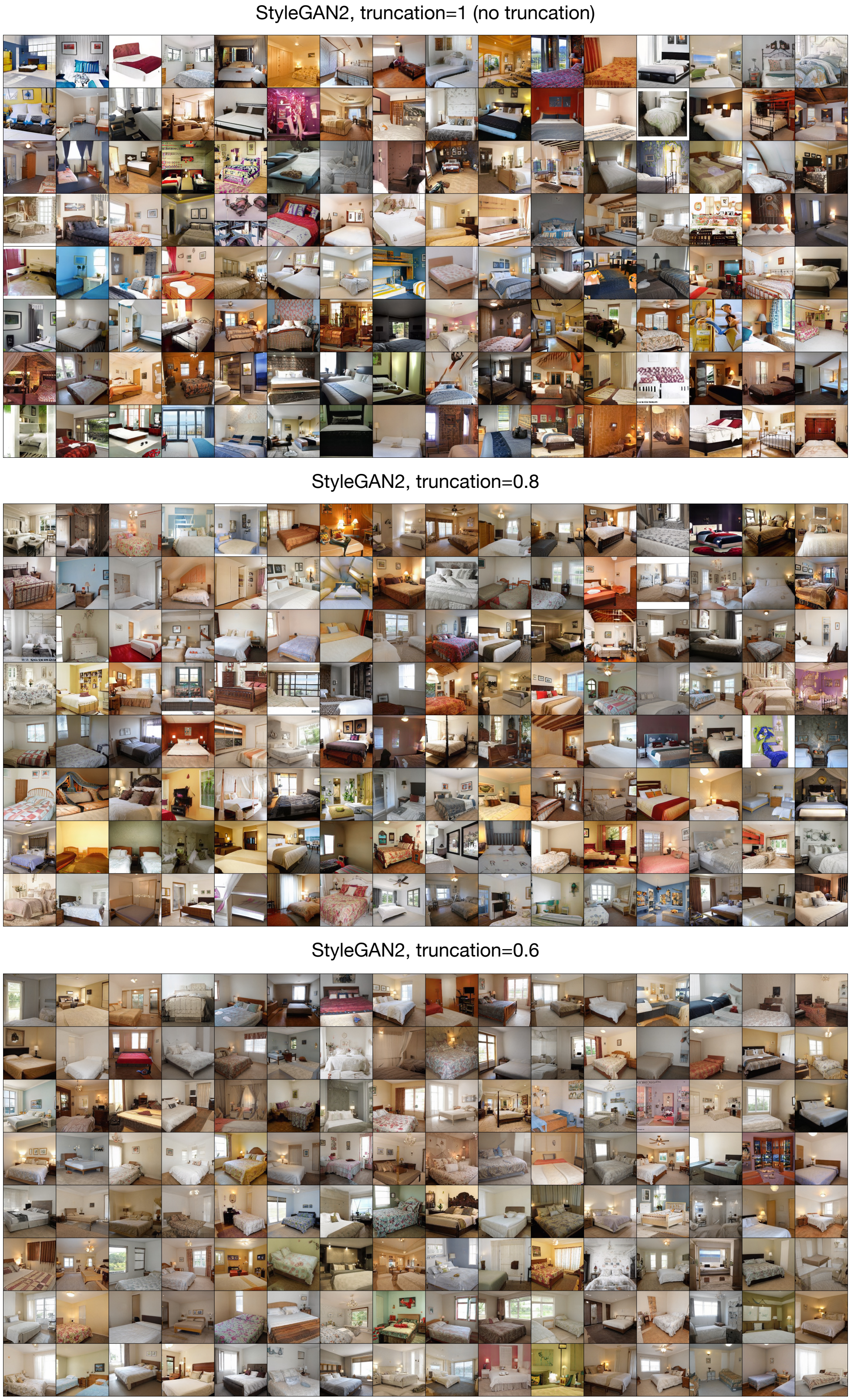


Figure 2.25: **Random samples:** We show uncurated random image samples from BlobGAN on LSUN bedrooms at various truncation levels. Please view zoomed in and in color for best results.



StyleGAN2, truncation=1 (no truncation)

StyleGAN2, truncation=0.8

StyleGAN2, truncation=0.6

Figure 2.26: **Random samples:** We show uncurated random image samples from StyleGAN2 on LSUN bedrooms at various truncation levels. Please view zoomed in and in color for best results.

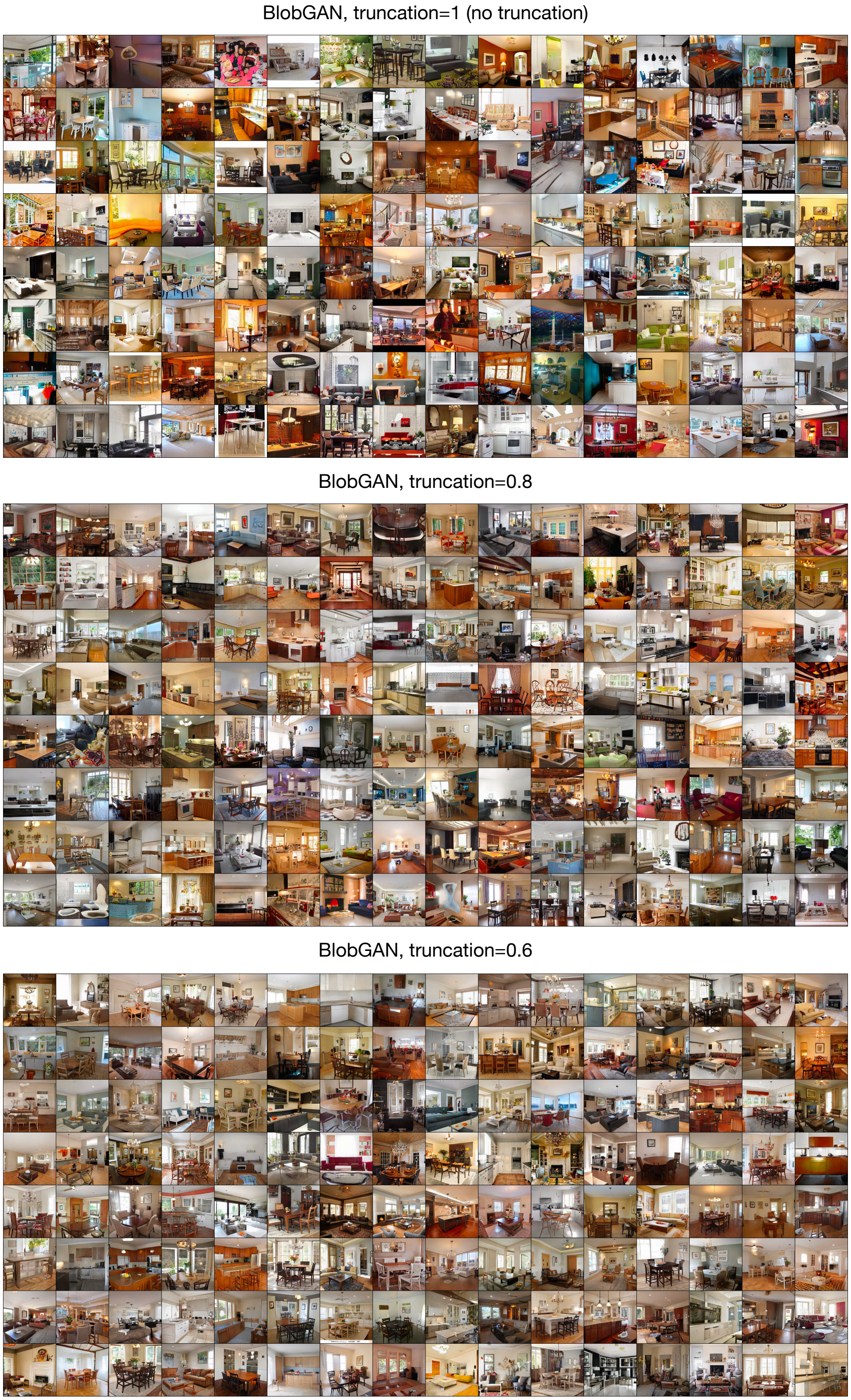
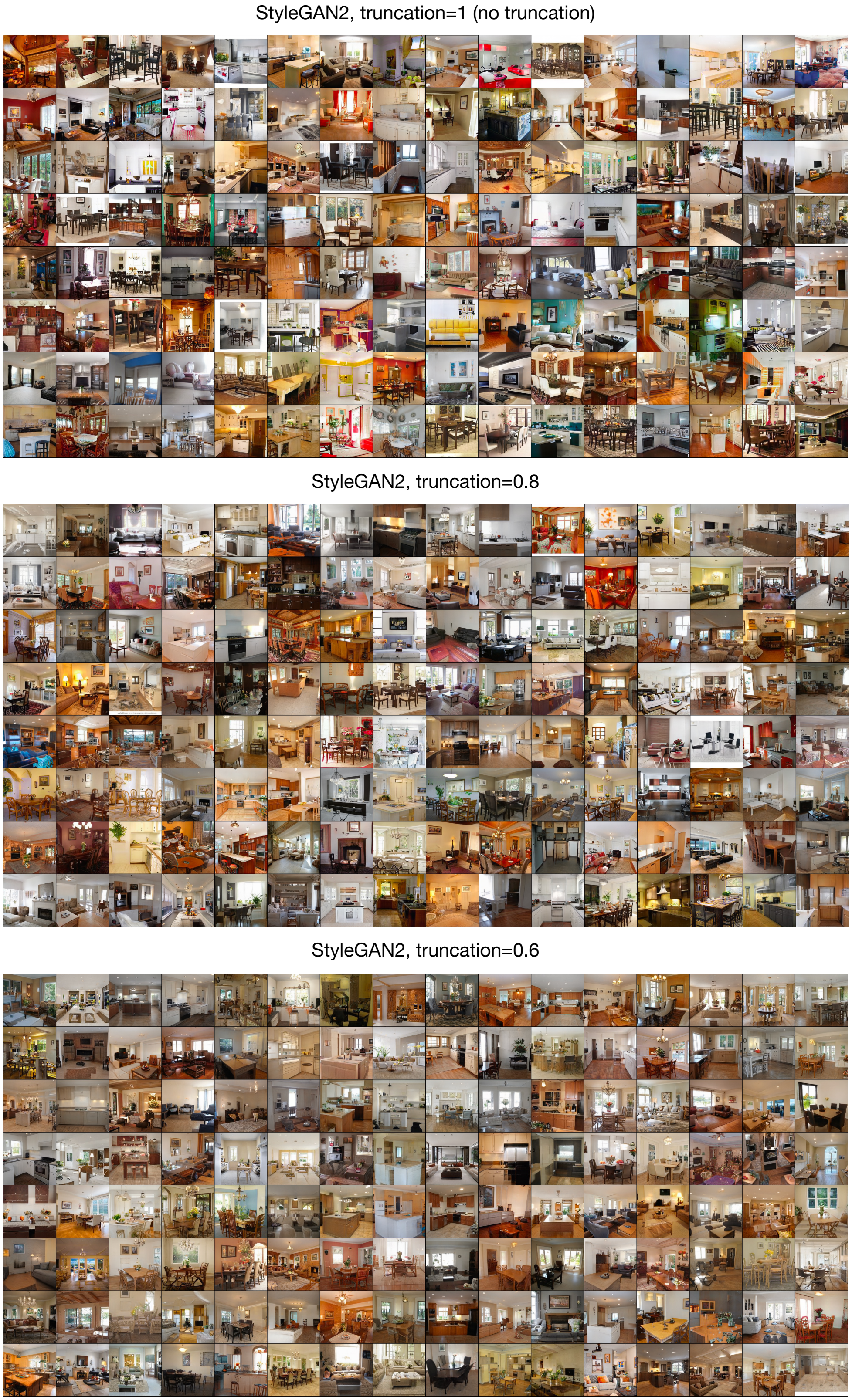


Figure 2.27: **Random samples:** We show uncurated random image samples from BlobGAN on LSUN kitchens, living rooms, and dining rooms at various truncation levels. Please view zoomed in and in color for best results.


StyleGAN2, truncation=1 (no truncation)

StyleGAN2, truncation=0.8

StyleGAN2, truncation=0.6

Figure 2.28: **Random samples:** We show uncurated random image samples from StyleGAN2 on LSUN kitchens, living rooms, and dining rooms at various truncation levels. Please view zoomed in and in color for best results.

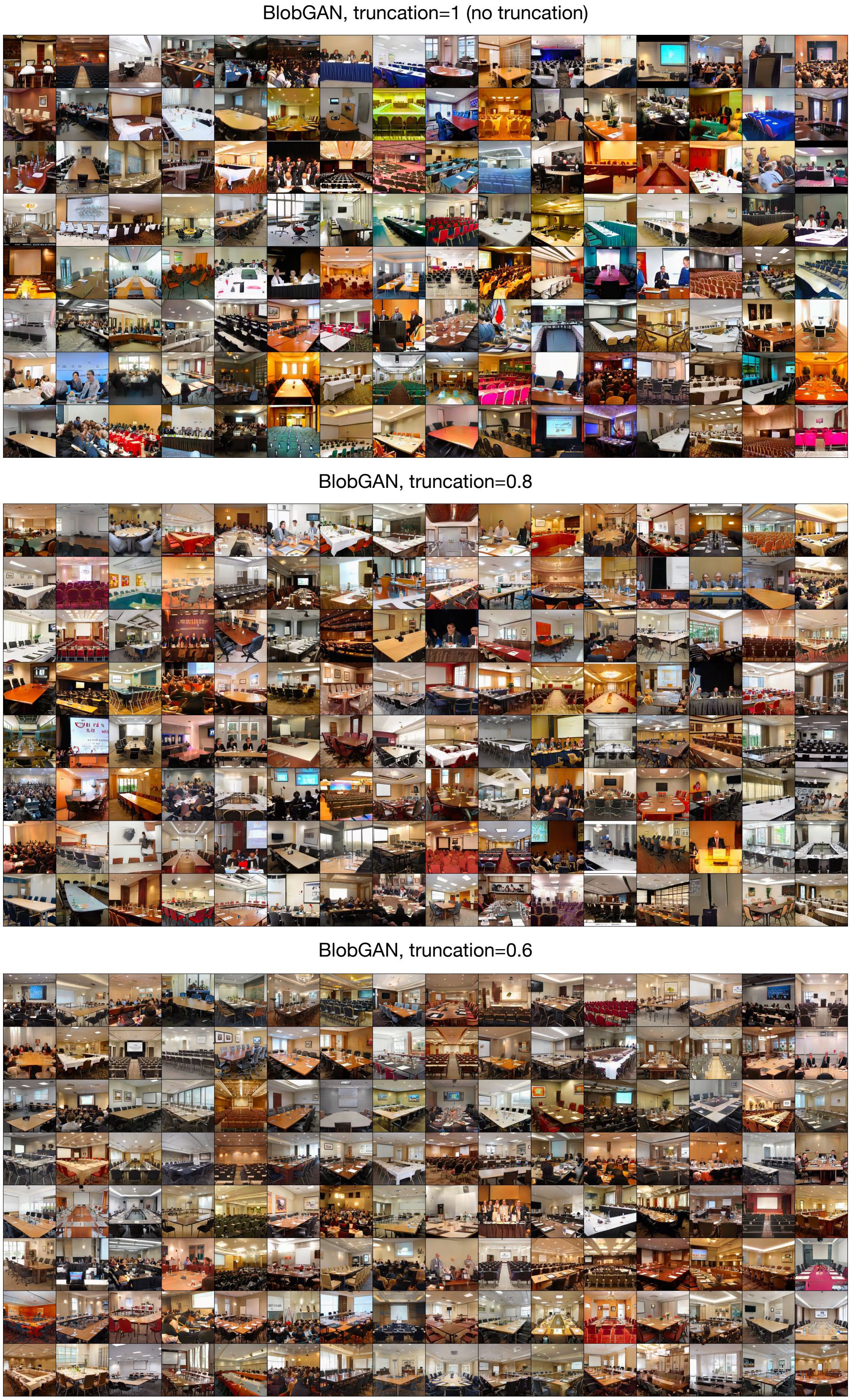


Figure 2.29: **Random samples:** We show uncurated random image samples from BlobGAN on LSUN conference rooms at various truncation levels. Please view zoomed in and in color for best results.

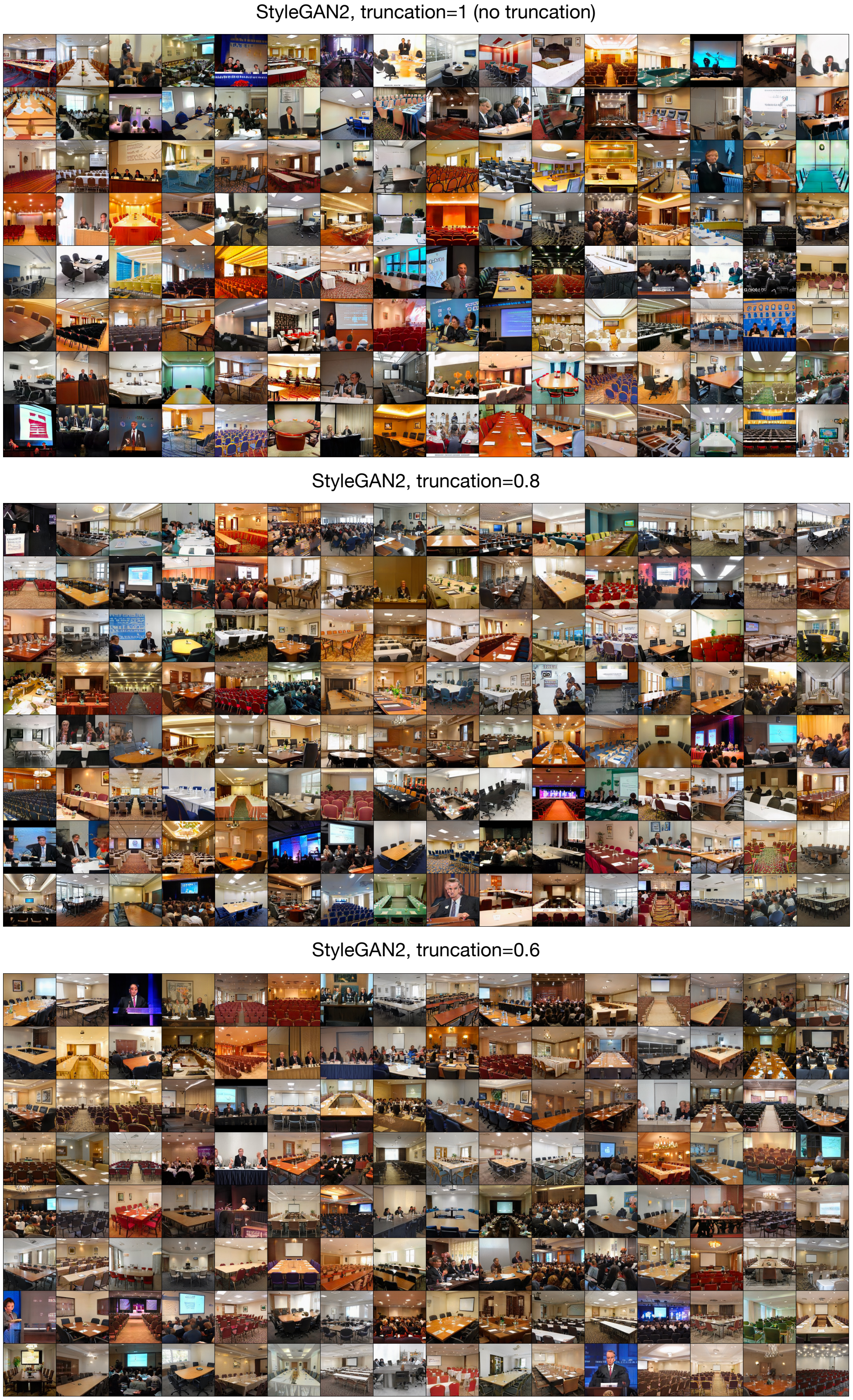

StyleGAN2, truncation=1 (no truncation)
StyleGAN2, truncation=0.8
StyleGAN2, truncation=0.6

Figure 2.30: **Random samples:** We show uncurated random image samples from StyleGAN2 on LSUN conference rooms at various truncation levels. Please view zoomed in and in color for best results.

Chapter 3

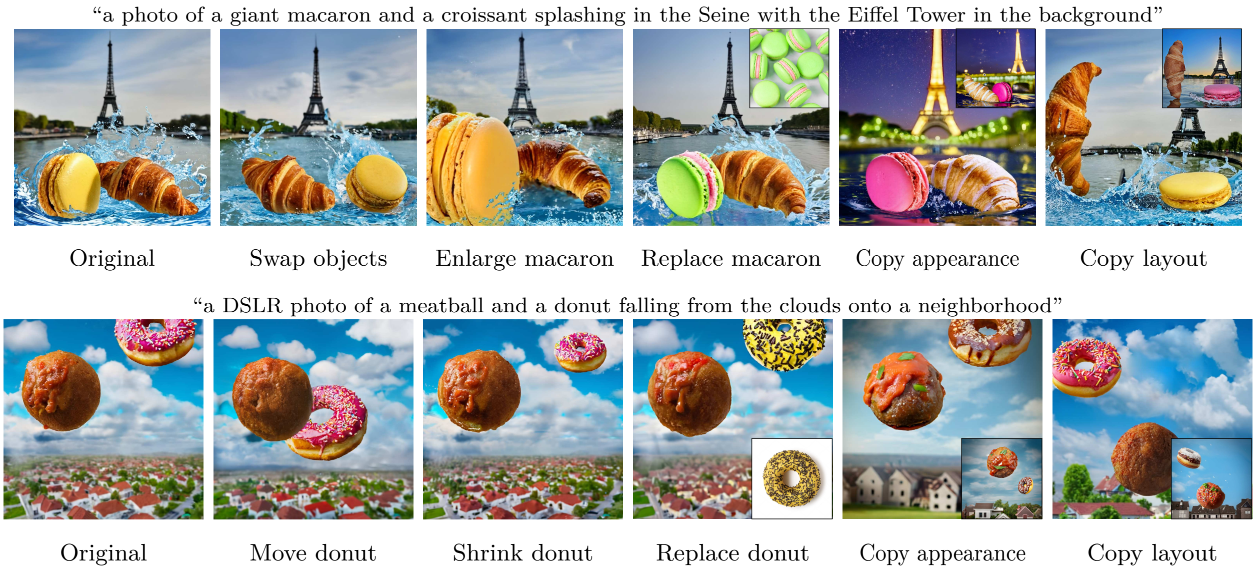
An Objective for Controllable Image Sampling

In contrast to models with strong representation bottlenecks such as those described in the previous chapter, large-scale generative models are capable of producing high-quality images from detailed text descriptions. However, many aspects of an image are difficult or impossible to convey through text. We introduce self-guidance, a method that provides greater control over generated images by guiding the internal representations of diffusion models. We demonstrate that properties such as the shape, location, and appearance of objects can be extracted from these representations and used to steer the sampling process. Self-guidance operates similarly to standard classifier guidance, but uses signals present in the pretrained model itself, requiring no additional models or training. We show how a simple set of properties can be composed to perform challenging image manipulations, such as modifying the position or size of specific objects, merging the appearance of objects in one image with the layout of another, composing objects from multiple images into one, and more. We also show that self-guidance can be used for editing real images.

3.1 Introduction

Generative image models have improved rapidly in recent years with the adoption of large text-image datasets and scalable architectures [196, 151, 137, 30, 143, 165, 55, 42]. These models are able to create realistic images given a text prompt describing just about anything. However, despite the incredible abilities of these systems, discovering the right prompt to generate the *exact* image a user has in mind can be surprisingly challenging. A key issue is that all desired aspects of an image must be communicated through text, even those that are difficult or even impossible to convey precisely.

To address this limitation, previous work has introduced methods [38, 147, 77, 96] that tune pretrained models to better control details that a user has in mind. These details are often supplied in the form of reference images along with a new textual prompt [19, 5] or other



"a photo of a giant macaron and a croissant splashing in the Seine with the Eiffel Tower in the background"

Original Swap objects Enlarge macaron Replace macaron Copy appearance Copy layout

"a DSLR photo of a meatball and a donut falling from the clouds onto a neighborhood"

Original Move donut Shrink donut Replace donut Copy appearance Copy layout

Figure 3.1: **Self-guidance** is a method for controllable image generation that guides sampling using the attention and activations of a pretrained diffusion model. With self-guidance, we can move or resize objects, or even replace them with items from real images, without changing the rest of the scene. We can also borrow the appearance of other images or rearrange scenes into new layouts.

forms of conditioning [202, 4, 150]. However, these approaches all either rely on fine-tuning with expensive paired data (thus limiting the scope of possible edits) or must undergo a costly optimization process to perform the few manipulations they are designed for. While some methods [52, 177, 104, 107] can perform zero-shot editing of an input image using a target caption describing the output, these methods only allow for limited control, often restricted to structure-preserving appearance manipulation or uncontrolled image-to-image translation.

By consequence, many simple edits still remain out of reach. For example, how can we move or resize one object in a scene without changing anything else? How can we take the appearance of an object in one image and copy it over to another, or combine the layout of one scene with the appearance of a second one? How can we generate images with certain items having precise shapes at specific positions on the canvas? This degree of control has been explored in the past in smaller scale settings [33, 21, 206, 97, 124, 195], but has not been convincingly demonstrated with modern large-scale diffusion models [151, 196, 135].

We propose *self-guidance*, a zero-shot approach which allows for direct control of the shape, position, and appearance of objects in generated images. Self-guidance leverages the rich representations learned by pretrained text-to-image diffusion models – namely, intermediate activations and attention – to steer attributes of entities and interactions between them. These constraints can be user-specified or transferred from other images, and rely only on knowledge internal to the diffusion model. Through a variety of challenging image manipulations, we demonstrate that self-guidance using only a few simple properties allows for granular, disentangled manipulation of the contents of generated images (Figure 3.1). Further, we show that self-guidance can also be used to reconstruct and edit real images.

Our key contributions are as follows:

- We introduce *self-guidance*, which takes advantage of the internal representations of pretrained text-to-image diffusion models to provide disentangled, zero-shot control over the generative process without requiring auxiliary models or supervision.
- We find that properties such as the size, location, shape, and appearance of objects can be extracted from these representations and used to meaningfully guide sampling in a zero-shot manner.
- We demonstrate that this small set of properties, when composed, allows for a wide variety of surprisingly complex image manipulations, including control of relationships between objects and the way modifiers bind to them.
- Finally, by reconstructing captioned images using their layout and appearance as computed by self-guidance, we show that we can extend our method to editing real images.

3.2 Background

3.2.1 Diffusion generative models

Diffusion models learn to transform random noise into high-resolution images through a sequential sampling process [164, 55, 166]. This sampling process aims to reverse a fixed time-dependent destructive process that corrupts data by adding noise. The learned component of a diffusion model is a neural network ϵ_θ that tries to estimate the denoised image, or equivalently the noise ϵ_t that was added to create the noisy image $z_t = \alpha_t x + \sigma_t \epsilon_t$. This network is trained with loss:

$$L(\theta) = \mathbb{E}_{t \sim \mathcal{U}(1,T), \epsilon_t \sim \mathcal{N}(0,\mathbf{I})} \Big[w(t) || \epsilon_t - \epsilon_\theta(z_t; t, y) ||^2 \Big], \tag{3.1}$$

where y is an additional conditioning signal like text, and $w(t)$ is a function weighing the contributions of denoising tasks to the training objective, commonly set to 1 [55, 81]. A common choice for ϵ_θ is a U-Net architecture with self- and cross-attention at multiple resolutions to attend to conditioning text in y [144, 151, 143]. Diffusion models are score-based models, where ϵ_θ can be seen as an estimate of the score function for the noisy marginal distributions: $\epsilon_\theta(z_t) \approx -\sigma_t \nabla_{z_t} \log p(z_t)$ [166].

Given a trained model, we can generate samples given conditioning y by starting from noise $z_T \sim \mathcal{N}(0, I)$, and then alternating between estimating the noise component and updating the noisy image:

$$\hat{\epsilon}_t = \epsilon_\theta(z_t; t, y), \quad z_{t-1} = \text{update}(z_t, \hat{\epsilon}_t, t, t-1, \epsilon_{t-1}), \tag{3.2}$$

where the update could be based on DDPM [55], DDIM [165], or another sampling method (see Appendix for details). Unfortunately, naïvely sampling from conditional diffusion models does not produce high-quality images that correspond well to the conditioning y. Instead, additional techniques are utilized to modify the sampling process by altering the update direction $\hat{\epsilon}_t$.

3.2.2 Guidance

A key capability of diffusion models is the ability to adapt outputs after training by *guiding* the sampling process. From the score-based perspective, we can think of guidance as composing score functions to sample from richer distributions or to introduce conditioning on auxiliary information [30, 94, 166]. In practice, using guidance involves altering the update direction $\hat{\epsilon}_t$ at each iteration.

Classifier guidance can generate conditional samples from an unconditional model by combining the unconditional score function for $p(z_t)$ with a classifier $p(y|z_t)$ to generate samples from $p(z_t|y) \propto p(y|z_t)p(z_t)$ [30, 166]. To use classifier guidance, one needs access to a labeled dataset and has to learn a noise-dependent classifier $p(y|z_t)$ that can be differentiated with respect to the noisy image z_t. While sampling, we can incorporate classifier guidance by modifying $\hat{\epsilon}_t$:

$$\hat{\epsilon}_t = \epsilon_\theta(z_t; t, y) - s\sigma_t \nabla_{z_t} \log p(y|z_t), \tag{3.3}$$

where s is an additional parameter controlling the guidance strength. Classifier guidance moves the sampling process towards images that are more likely according to the classifier [30], achieving a similar effect to truncation in GANs [16], and can also be applied with pretrained classifiers by first denoising the intermediate noisy image (though this requires additional approximations [4]).

In general, we can use any energy function $g(z_t; t, y)$ to guide the diffusion sampling process, not just the probabilities from a classifier. g could be the approximate energy from another model [94], a similarity score from a CLIP model [112], an arbitrary time-independent energy as in universal guidance [4], bounding box penalties on attention [23], or any attributes of the noisy images. We can incorporate this additional guidance alongside *classifier-free guidance* [56] to obtain high-quality text-to-image samples that also have low energy according to g:

$$\hat{\epsilon}_t = (1+s)\epsilon_\theta(z_t; t, y) - s\epsilon_\theta(z_t; t, \emptyset) + v\sigma_t \nabla_{z_t} g(z_t; t, y), \tag{3.4}$$

where s is the classifier-free guidance strength and v is an additional guidance weight for g. As with classifier guidance, we scale by σ_t to convert the score function to a prediction of ϵ_t. The main contribution of our work is to identify energy functions g useful for controlling properties of objects and interactions between them.

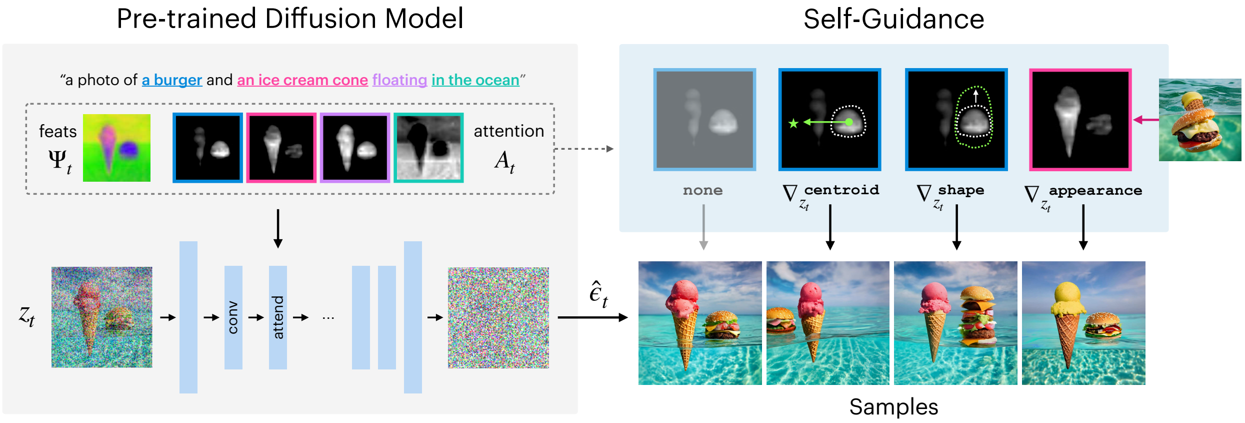


Figure 3.2: **Overview:** We leverage representations learned by text-image diffusion models to steer generation with *self-guidance*. By constraining intermediate activations Ψ_t and attention interactions $\mathcal{A}_t$, self-guidance can control properties of entities named in the prompt. For example, we can change the position and shape of the burger, or copy the appearance of ice cream from a source image.

3.2.3 Where can we find signal for controlling diffusion?

While guidance is a flexible way of controlling the sampling process, energy functions typically used [202, 4] require auxiliary models (adapted to be noise-dependent) as well as data annotated with properties we would like to control. Can we circumvent these costs? Recent work [52, 177] has shown that the intermediate outputs of the diffusion U-Net encode valuable information [83, 132] about the structure and content of the generated images. In particular, the self and cross-attention maps $\left\{\mathcal{A}_{i,t} \in \mathbb{R}^{H_i \times W_i \times K}\right\}$ often encode structural information [52] about object position and shape, while the network activations $\left\{\Psi_{i,t} \in \mathbb{R}^{H_i \times W_i \times D_i}\right\}$ allow for maintaining coarse appearance [177] when extracted from appropriate layers. While these editing approaches typically share attention and activations naively between subsequent sampling passes, drastically limiting the scope of possible manipulations, we ask: what if we tried to harness model internals in a more nuanced way?

3.3 Self-guidance

Inspired by the rich representations learned by diffusion models, we propose self-guidance, which places constraints on intermediate activations and attention maps to steer the sampling process and control entities named in text prompts (see Fig. 3.2). These constraints can be user-specified or copied from existing images, and rely only on knowledge internal to the diffusion model.

We identify a number of properties useful for meaningfully controlling generated images, derived from the set of softmax-normalized attention matrices $\left\{\mathcal{A}_{i,t} \in \mathbb{R}^{H_i \times W_i \times K}\right\}$ and activations $\left\{\Psi_{i,t} \in \mathbb{R}^{H_i \times W_i \times D_i}\right\}$ extracted from the standard denoising forward pass $\epsilon_\theta(z_t; t, y)$. To control an object mentioned in the text conditioning y at token indices k, we can manipulate the corresponding attention channel(s) $\mathcal{A}_{i,t,\cdot,\cdot,k} \in \mathbb{R}^{H_i \times W_i \times |k|}$ and activations $\Psi_{i,t}$ (extracted at timestep t from a noisy image z_t given text conditioning y) by adding guidance terms to Eqn. 3.4.

Object position. To represent the position of an object (omitting attention layer index and timestep for conciseness), we find the center of mass of each relevant attention channel:

$$\texttt{centroid}\,(k) = \frac{1}{\sum_{h,w} \mathcal{A}_{h,w,k}} \begin{bmatrix} \sum_{h,w} w \cdot \mathcal{A}_{h,w,k} \\ \sum_{h,w} h \cdot \mathcal{A}_{h,w,k} \end{bmatrix} \tag{3.5}$$

We can use this property to guide an object to an absolute target position on the image. For example, to move "$\texttt{burger}$" to position $(0.3, 0.5)$, we can minimize $\|(0.3, 0.5) - \texttt{centroid}\,(k)\,\|_1$. We can also perform a relative transformation, e.g., move "$\texttt{burger}$" to the right by $(0.1, 0.0)$ by minimizing $\|\texttt{centroid}_{\text{orig}}\,(k) + (0.1, 0.0) - \texttt{centroid}\,(k)\,\|_1$.

Object size. To compute an object's size, we spatially sum its corresponding attention channel:

$$\texttt{size}\,(k) = \frac{1}{HW} \sum_{h,w} \mathcal{A}_{h,w,k} \tag{3.6}$$

In practice, we find it beneficial to differentiably threshold the attention map $\mathcal{A}_{\text{thresh}}$ before computing its size, to eliminate the effect of background noise. We do this by taking a soft threshold at the midpoint of the per-channel minimum and maximum values (see Appendix for details). As with position, one can guide to an absolute size (*e.g.* half the canvas) or a relative one (*e.g.* 10% larger).

Object shape. For even more granular control than position and size, we can represent the object's exact shape directly through the thresholded attention map itself:

$$\texttt{shape}(k) = \mathcal{A}_k^{\text{thresh}} \tag{3.7}$$

This shape can then be guided to match a specified binary mask (either provided by a user or extracted from the attention from another image) with $\|\texttt{target\_shape} - \texttt{shape}\,(k)\,\|_1$. Note that we can apply any arbitrary transformation (scale, rotation, translation) to this shape before using it as a guidance target, which allows us to manipulate objects while maintaining their silhouette.

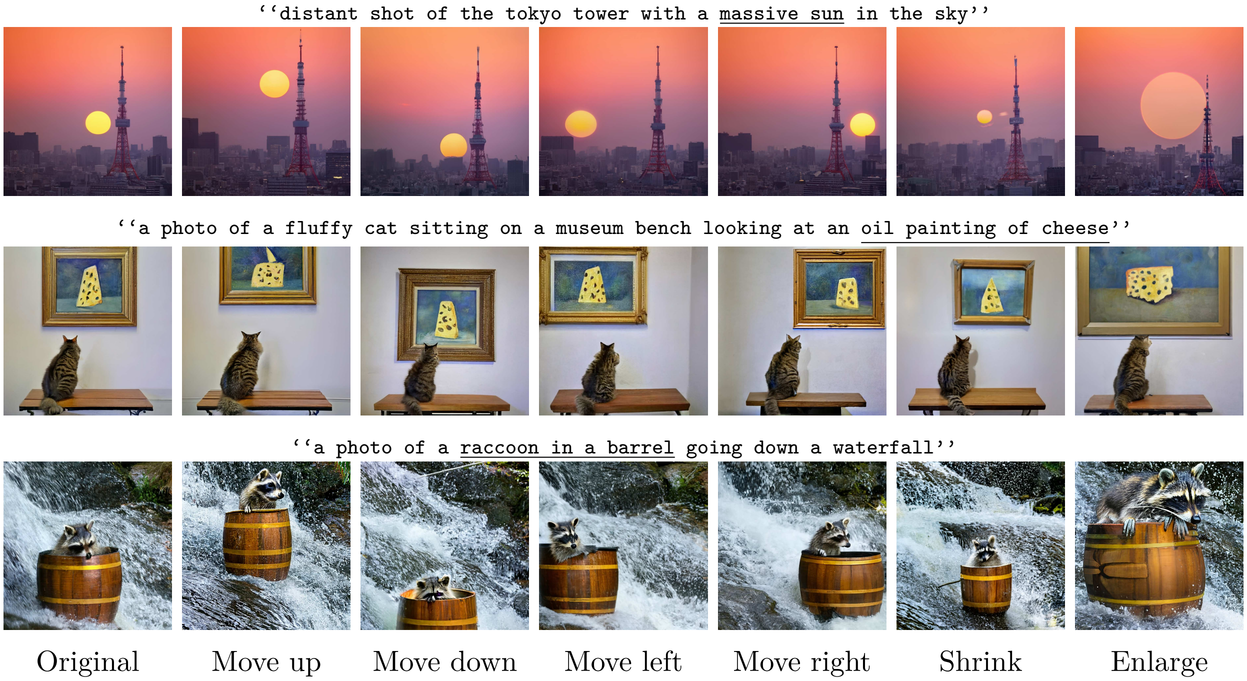

''distant shot of the tokyo tower with a massive sun in the sky''
''a photo of a fluffy cat sitting on a museum bench looking at an oil painting of cheese''
''a photo of a raccoon in a barrel going down a waterfall''
Original
Move up
Move down
Move left
Move right
Shrink
Enlarge

Figure 3.3: **Moving and resizing objects.** By only changing the properties of one object (as in Eqn. 3.9), we can move or resize that object without modifying the rest of the image. In these examples, we modify "`massive sun`", "`oil painting of cheese`", and "`raccoon in a barrel`", respectively.

Object appearance. Considering thresholded attention a rough proxy for object extent, and spatial activation maps as representing local appearance (since they ultimately must be decoded into an unnoised RGB image), we can reach a notion of object-level appearance by combining the two:

$$\texttt{appearance}(k) = \frac{\sum_{h,w} \texttt{shape}(k) \odot \Psi}{\sum_{h,w} \texttt{shape}(k)} \tag{3.8}$$

3.4 Composing self-guidance properties

The small set of properties introduced in Section 3.3 can be composed to perform a wide range of image manipulations, including many that are intractable through text. We showcase this collection of manipulations and, when possible, compare to prior work that accomplishes similar effects. All experiments were performed on Imagen [151], producing 1024 × 1024 samples. For more samples and details on the implementation of self-guidance, please see the Appendix.

Adjusting specific properties. By guiding one property to change and all others to keep their original values, we can modify single objects in isolation (Fig. 3.3b-3.3e). For a caption $C = y$ with words at indices $\{c_i\}$, in which $O = \{o_j\} \subseteq C$ are objects, we can move an object o_k at time t with:

$$\begin{aligned} g = w_0 & \overbrace{\frac{1}{|O|-1}\sum_{o \neq o_k \in O} \frac{1}{|\mathcal{A}|}\sum_{i=0}^{|\mathcal{A}|} \|\texttt{shape}_{i,t,\text{orig}}(o) - \texttt{shape}_{i,t}(o)\|_1}^{\text{Fix all other object shapes}} \\ + w_1 & \overbrace{\frac{1}{|O|}\sum_{o \in O} \|\texttt{appearance}_{t,\text{orig}}(o) - \texttt{appearance}_t(o)\|_1}^{\text{Fix all appearances}} \\ + w_2 & \overbrace{\frac{1}{|\mathcal{A}|}\sum_{i=0}^{|\mathcal{A}|} \|\mathcal{T}\left(\texttt{shape}_{i,t,\text{orig}}(o_k)\right) - \texttt{shape}_{i,t}(o_k)\|_1}^{\text{Guide } o_k\text{'s shape to translated original shape}} \end{aligned} \tag{3.9}$$

Where $\texttt{shape}_{\text{orig}}$ and $\texttt{shape}$ are extracted from the generation of the initial and edited image, respectively. Critically, $\mathcal{T}$ lets us define whatever transformation of the $H_i \times W_i$ spatial attention map we want. To move an object, $\mathcal{T}$ translates the attention mask the desired amount. We can also resize objects (Fig. 3.3f-3.3g) with Eqn. 3.9 by changing $\mathcal{T}$ to up- or down-sample shape matrices.

Constraining per-object layout but not appearance finds new "styles" for the same scene (Fig. 3.4):

$$g = w_0 \overbrace{\frac{1}{|O|}\sum_{o \in O} \frac{1}{|\mathcal{A}|}\sum_{i=0}^{|\mathcal{A}|} \|\texttt{shape}_{i,t,\text{orig}}(o) - \texttt{shape}_{i,t}(o)\|_1}^{\text{Fix all object shapes}} \tag{3.10}$$

We can alternatively choose to guide all words, not just nouns or objects, changing summands to $\sum_{c \neq o_k \in C}$ instead of $\sum_{c \neq o_k \in O}$. See Appendix for further discussion.

Composition between images. We can compose properties across multiple images into a cohesive sample, *e.g.* the layout of an image A with the appearance of objects in another image B (Fig. 3.5):

$$\begin{aligned} g = w_0 & \overbrace{\frac{1}{|O|}\sum_{o \in O} \frac{1}{|\mathcal{A}|}\sum_{i=0}^{|\mathcal{A}|} \|\texttt{shape}_{i,t,A}(o) - \texttt{shape}_{i,t}(o)\|_1}^{\text{Copy object shapes from A}} \\ + w_1 & \overbrace{\frac{1}{|O|}\sum_{o \in O} \|\texttt{appearance}_{t,B}(o) - \texttt{appearance}_t(o)\|_1}^{\text{Copy object appearance from B}} \end{aligned} \tag{3.11}$$

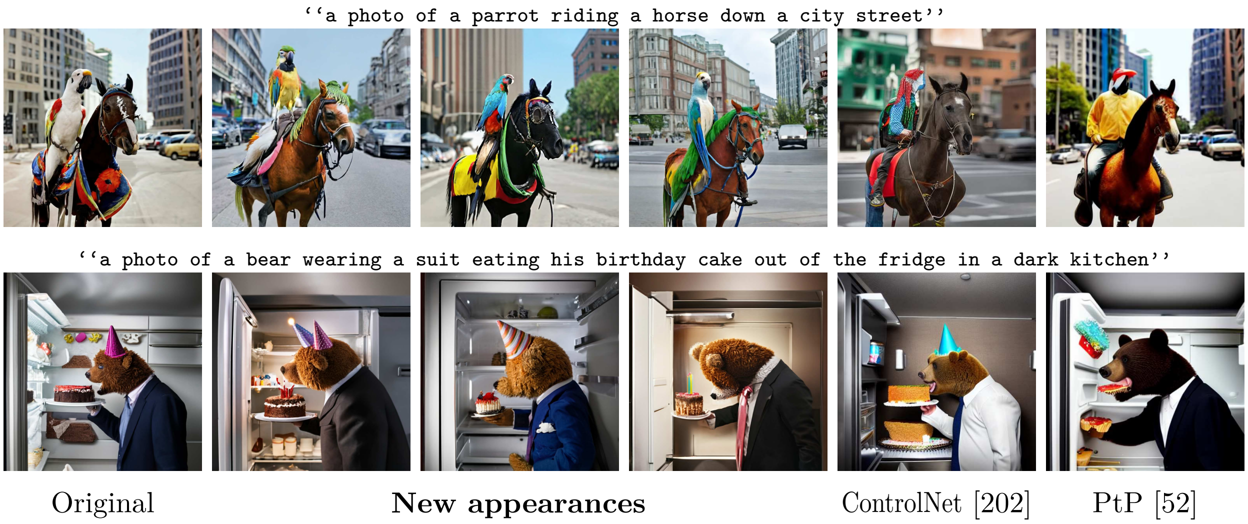

‘‘a photo of a parrot riding a horse down a city street’’
‘‘a photo of a bear wearing a suit eating his birthday cake out of the fridge in a dark kitchen’’
Original
New appearances
ControlNet [202]
PtP [52]

Figure 3.4: **Sampling new appearances.** By guiding object shapes (Eqn. 3.7) towards reconstruction of a given image's layout (left), we can sample new appearances for a given scene (right). We compare to ControlNet v1.1-Depth [202] and Prompt-to-Prompt [52], which accomplish a similar objective.

We can also borrow only appearances, dropping the first term to sample new arrangements for the same objects, as in the last two columns of Figure 3.5.

Highlighting the compositionality of self-guidance terms, we can further inherit the appearance and/or shape of objects from several images and combine them into one (Fig. 3.6). Say we have J images, where we are interested in keeping a single object o_{k_j} from each one. We can collage these objects "in-place" – *i.e.* maintaining their shape, size, position, and appearance – straightforwardly:

$$\begin{aligned} g = w_0 & \overbrace{\frac{1}{J}\sum_j \frac{1}{|\mathcal{A}|}\sum_{i=0}^{|\mathcal{A}|} \|\texttt{shape}_{i,t,j}(o_{k_j}) - \texttt{shape}_{i,t}(o_k)\|_1}^{\text{Copy each object's shape, position, and size}} \\ & + w_1 \overbrace{\frac{1}{J}\sum_j \|\texttt{appearance}_{t,j}(o_{k_j}) - \texttt{appearance}_t(o_k)\|_1}^{\text{Copy each object's appearance}} \end{aligned} \tag{3.12}$$

We can also take only the appearances of the objects from these images and copy the layout from another image, useful if object positions in the J images are not mutually compatible (Fig. 3.6f).

Editing with real images. Our approach is not limited to only images generated by a model, whose internals we have access to by definition. By running T noised versions of a

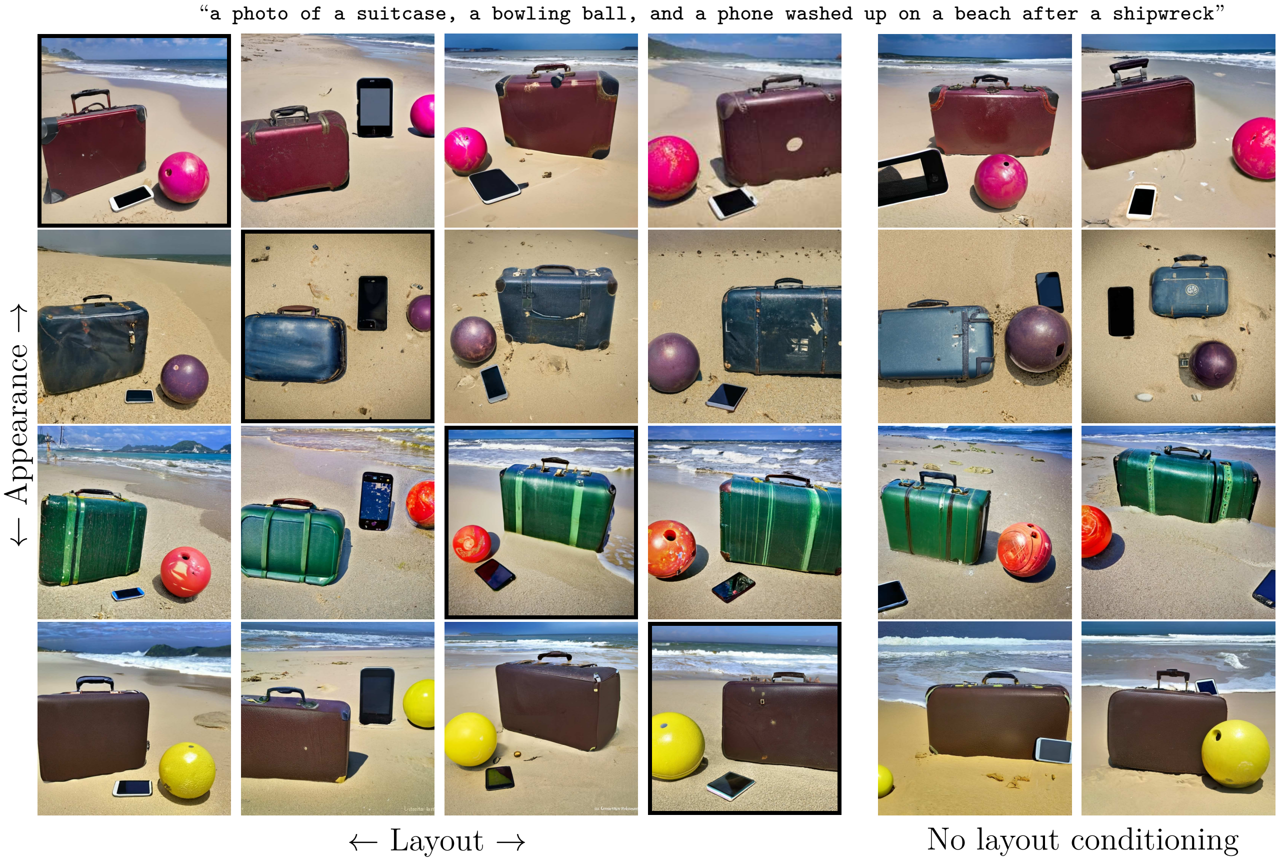
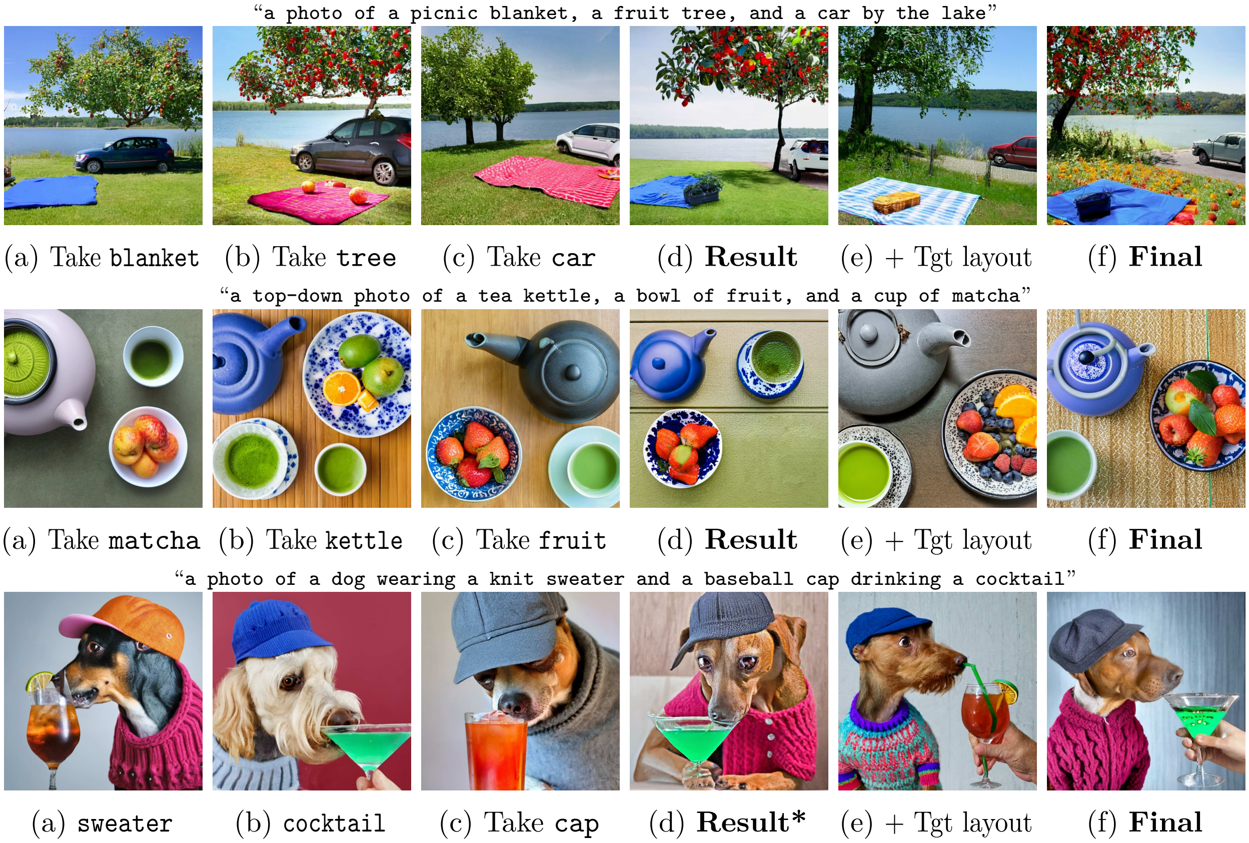

"a photo of a suitcase, a bowling ball, and a phone washed up on a beach after a shipwreck"
← Appearance →
← Layout →
No layout conditioning

Figure 3.5: **Mix-and-match.** By guiding samples to take object shapes from one image and appearance from another (Eqn. 3.11), we can rearrange images into layouts from other scenes. Input images are along the diagonal. We can also sample new layouts of a scene by only guiding appearance (right).

(captioned) existing image through a denoiser – one for each forward process timestep – we extract a set of intermediates that can be treated as if it came from a reverse sampling process (see Appendix for more details). In Fig. 3.8, we show that, by guiding shape and appearance for all tokens, we generate faithful reconstructions of real images. More importantly, we can manipulate these real images just as we can generated ones, successfully controlling properties such as appearance, position, or size. We can also transfer the appearance of an object of interest into new contexts (Fig. 3.7), from only one source image, and without any fine-tuning:

$$g = w_0 \overbrace{\left\|\texttt{appearance}_{t,\text{orig}}(o_{k_{\text{orig}}}) - \texttt{appearance}_{t}\left(o_k\right)\right\|_1}^{\text{Copy object appearance}} \tag{3.13}$$

Attributes and interactions. So far we have focused only on the manipulation of objects, but we can apply our method to any concept in the image, as long as it appears in the caption. We demonstrate manipulation of verbs and adjectives in Fig. 3.9, and show an



"a photo of a picnic blanket, a fruit tree, and a car by the lake"

(a) Take blanket (b) Take tree (c) Take car (d) **Result** (e) + Tgt layout (f) **Final**

"a top-down photo of a tea kettle, a bowl of fruit, and a cup of matcha"

(a) Take matcha (b) Take kettle (c) Take fruit (d) **Result** (e) + Tgt layout (f) **Final**

"a photo of a dog wearing a knit sweater and a baseball cap drinking a cocktail"

(a) sweater (b) cocktail (c) Take cap (d) **Result*** (e) + Tgt layout (f) **Final**

Figure 3.6: **Compositional generation.** A new scene (d) can be created by collaging objects from multiple images (Eqn. 3.12). Alternatively – *e.g.* if objects cannot be combined at their original locations due to incompatibilities in these images' layouts (*as in the bottom row) – we can borrow only their appearance, and specify layout with a new image (e) to produce a composition (f) (Eqn. 3.19).

example where certain self-guidance constraints can help in enforcing attribute binding in the generation process.

3.5 Discussion

We introduce a method for guiding the diffusion sampling process to satisfy properties derived from the attention maps and activations within the denoising model itself. While we propose a number of such properties, many more certainly exist, as do alternative formulations of those presented in this paper. Among the proposed collection of properties, a few limitations stand out.

The reliance on cross-attention maps imposes restrictions by construction, precluding

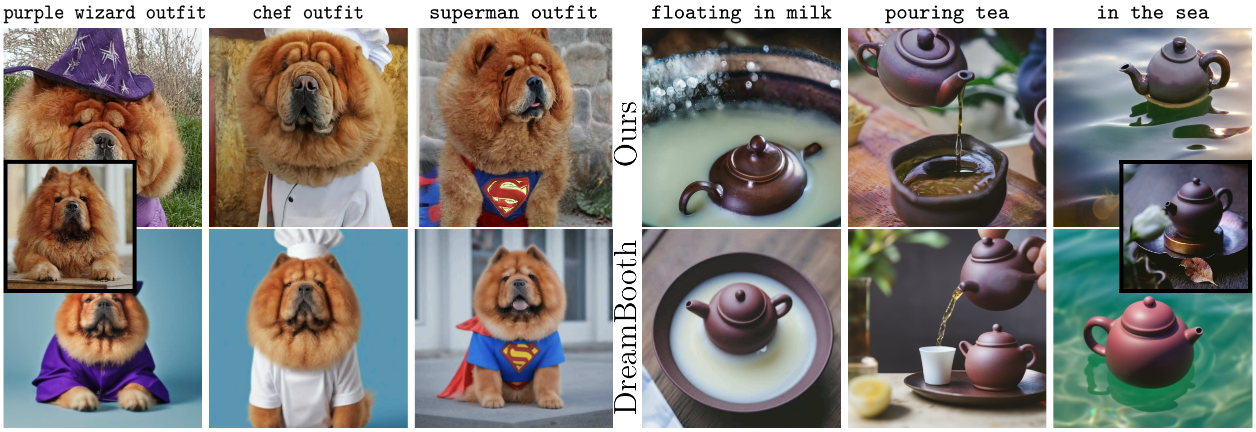


Figure 3.7: **Appearance transfer from real images.** By guiding the appearance of a generated object to match that of one in a real image (outlined) as in Eqn. 3.13, we can create scenes depicting an object from real life, similar to DreamBooth [147], but *without any fine-tuning and only using one image.*

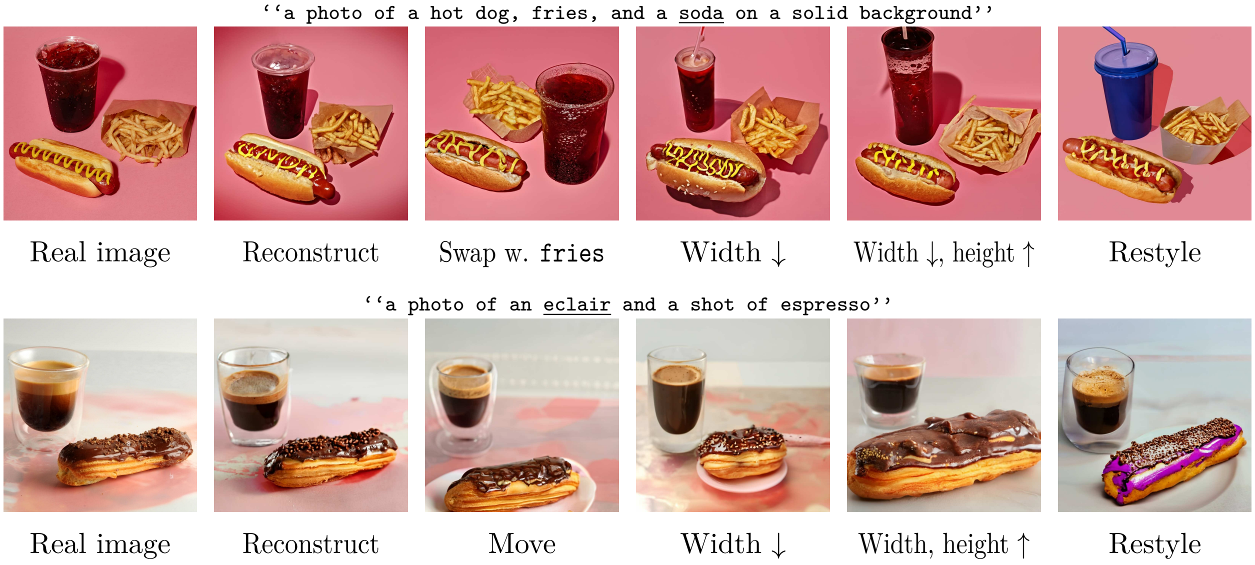


Figure 3.8: **Real image editing.** Our method enables the spatial manipulation of objects (shown in Figure 3.3 for generated images) for *real* images as well.

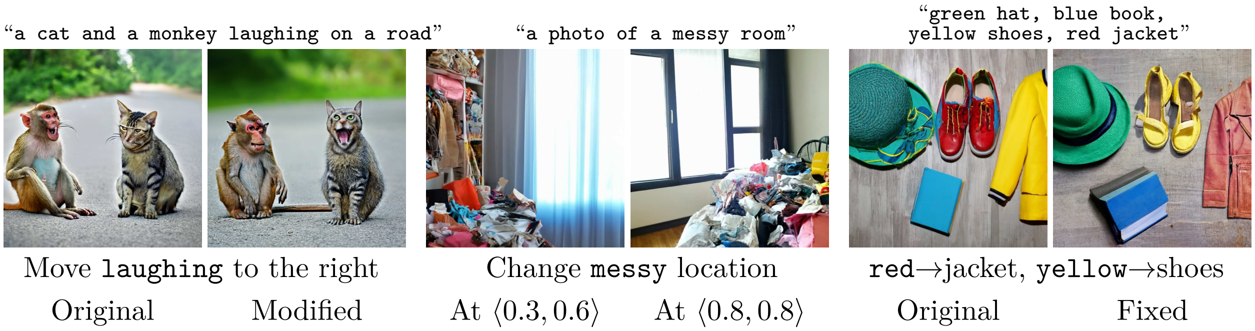

"a cat and a monkey laughing on a road"
"a photo of a messy room"
"green hat, blue book, yellow shoes, red jacket"
Move laughing to the right
Original
Modified
Change messy location
At ⟨0.3, 0.6⟩
At ⟨0.8, 0.8⟩
red→jacket, yellow→shoes
Original
Fixed

Figure 3.9: **Manipulating non-objects.** The properties of any word in the input prompt can be manipulated, not only nouns. Here, we show examples of relocating adjectives and verbs. The last example shows a case in which additional self-guidance can correct improper attribute binding.

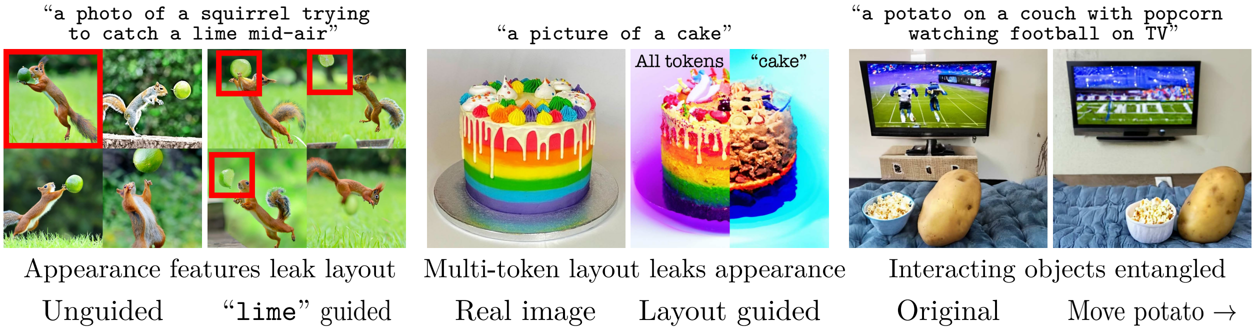

"a photo of a squirrel trying to catch a lime mid-air"
"a picture of a cake"
"a potato on a couch with popcorn watching football on TV"
All tokens
"cake"
Appearance features leak layout
Unguided
"lime" guided
Multi-token layout leaks appearance
Real image
Layout guided
Interacting objects entangled
Original
Move potato →

Figure 3.10: **Limitations.** Setting high guidance weights for appearance terms tends to introduce unwanted leakage of object position. Similarly, while heavily guiding the shape of one word simply matches that object's layout as expected, high guidance on the shapes of all tokens results in a leak of appearance information. Finally, in some cases, objects are entangled in attention space, making it difficult to control them independently.

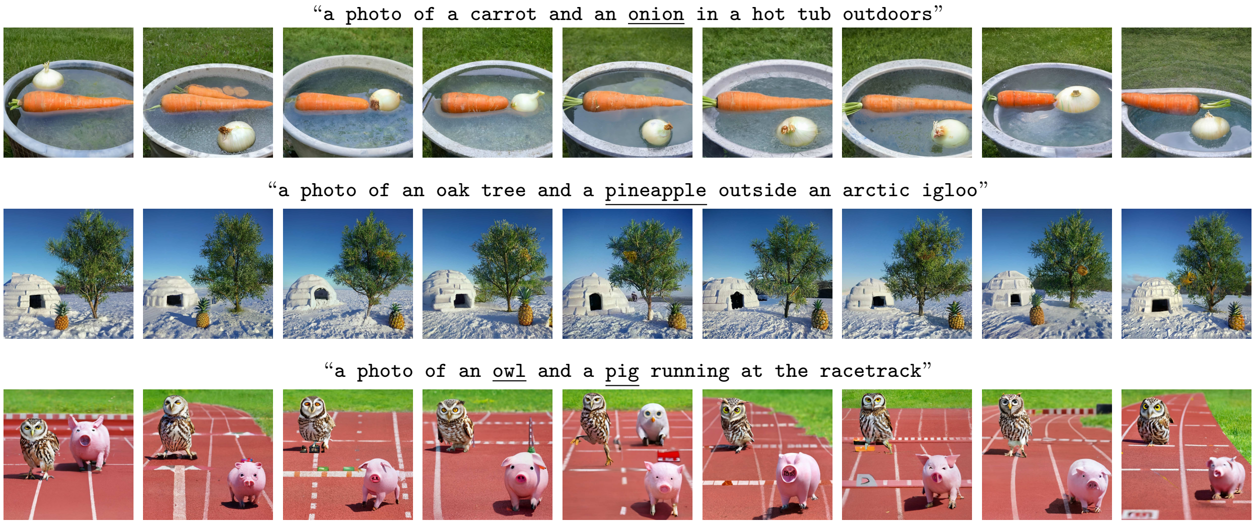
control over any object that is not described in the conditioning text prompt and hindering fully disentangled control between interacting objects due to correlations in attention maps (Fig. 3.10e-3.10f). Selectively applying attention guidance at certain layers or timesteps may result in more effective disentanglement.

Our experiments also show that appearance features often contain undesirable information about spatial layout (Fig. 3.10a-3.10b), perhaps since the model has access to positional information in its architecture. The reverse is also sometimes true: guiding the shape of multiple tokens occasionally betrays the appearance of an object (Fig. 3.10c-3.10d), implying that hidden high-frequency patterns arising from interaction between attention channels may be used to encode appearance. These findings suggest that our method could serve as a window into the inner workings of diffusion models and provide valuable experimental evidence to inform future research.

3.6 Appendix

3.6.1 Implementation details

We apply our self-guidance term following best practices for classifier-free guidance on Imagen [151]. Specifically, where N is the number of DDPM steps, we take the first $\frac{3N}{16}$ steps with self-guidance and the last $\frac{N}{32}$ without. The remaining $\frac{25N}{32}$ steps are alternated between using


"a photo of a carrot and an onion in a hot tub outdoors"
"a photo of an oak tree and a pineapple outside an arctic igloo"
"a photo of an owl and a pig running at the racetrack"

Original

(b) Edited

Figure 3.11: **Moving objects.** Non-cherry-picked results for moving objects in scenes using Eqn. 3.9. We move `onion` down and to the right, `pineapple` to the right, and `owl` up and `pig` down, respectively. All scenes use weights $w_0 = 1.5$, $w_1 = 0.25$, and $w_2 = 2$.

self-guidance and not using it. We use $N = 1024$ steps. Our method works with 256 and 512 steps as well, though self-guidance weights occasionally require adjustment. We set $v = 7500$ in Eqn. 3.4 as an overall scale for gradients of the functions g defined below — we find that the magnitude of per-pixel gradients is quite small (often in the range of 10^{-7} to 10^{-6}, so such a high weight is needed to induce changes.

We apply `centroid`, `size`, and `shape` terms on *all cross-attention interactions* in the model we use. In total, there are 36 of these, across the encoder, bottleneck, and decoder, at 8×8, 16×16, and 32×32 resolutions. We apply the `appearance` term using the activations of the penultimate layer in the decoder (two layers before the prediction readout) and the final cross-attention operation.We experimented with features from other parts of the U-Net denoiser, namely early in the encoder before positional information can propagate through the image (to prevent appearance-layout entanglement), but found these to work significantly worse. To avoid degenerate solutions, we apply a stop-gradient to the attention in the `appearance` term so only information about activations is back-propagated. We take the mean spatially of all `shape` terms and across activation dimensions for all `appearance` terms, which we omit in all equations for conciseness.

Attention mask binarization. In practice, it is beneficial to differentiably binarize the attention map (with sharpness controlled by s) before computing its size or utilizing its shape, to eliminate the effect of background noise (this is empirically less important when guiding centroids, so we do not binarize in that case). We do this by taking a soft threshold at the midpoint of the per-channel minimum and maximum values. More specifically, we apply a shifted sigmoid on the attention normalized to have minimum 0 and maximum 1, followed by another such normalization to ensure the high value is 1 and the low 0 after applying the sigmoid. We use $s = 10$ and redefine Eqn. 3.6.

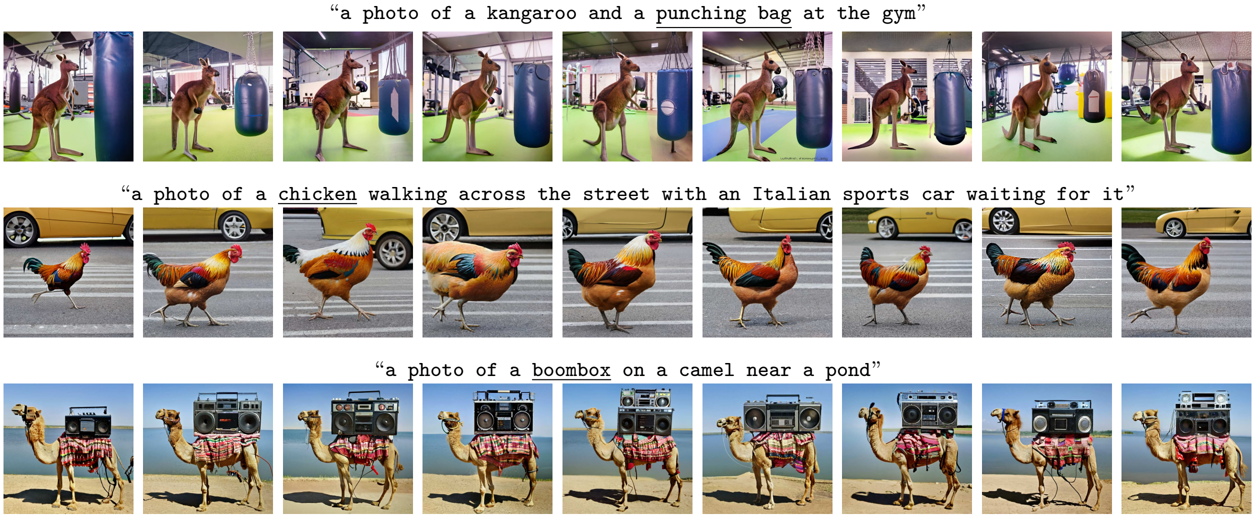
$$\texttt{normalize}(\mathbf{X}) = \frac{\mathbf{X} - \min_{h,w}(\mathbf{X})}{\max_{h,w}(\mathbf{X}) - \min_{h,w}(\mathbf{X})} \tag{3.14}$$

$$\mathcal{A}^{\text{thresh}} = \texttt{normalize}\left(\texttt{sigmoid}\left(s \cdot \left(\texttt{normalize}(\mathcal{A}) - 0.5\right)\right)\right) \tag{3.15}$$

$$\texttt{size}\,(k) = \frac{1}{HW}\sum_{h,w} \mathcal{A}^{\text{thresh}}_{h,w,k} \tag{3.16}$$

3.6.2 Using self-guidance

Maximizing consistency. In general, we find that sharing the same sequence of noise in the DDPM process between an image and its edited version is *not necessary* to maintain high levels of consistency, but can help if extreme precision is desired. We find that maintaining object silhouettes under transformations such as resizing and repositioning is more effective if applying a transformation $\mathcal{T}$ to the original `shape`, rather than expressing the same change through `centroid` and `size`.


“a photo of a kangaroo and a punching bag at the gym”
“a photo of a chicken walking across the street with an Italian sports car waiting for it”
“a photo of a boombox on a camel near a pond”

Original (b) Edited

Figure 3.12: **Resizing objects.** Non-cherry-picked results for resizing objects in scenes using Eqn. 3.9, with $\mathcal{T}$ specified to up- or down-sample attention maps. We reduce the `punching bag`'s height 0.5× and enlarge `chicken` 2.5× and `boombox` 2×. All scenes use $w_0 = 2$, $w_1 = 0.25$, and $w_2 = 3$.

Guiding “background” words. To keep all objects of the scene fixed but one (Fig. 3.3), one can either guide all other tokens in the prompt (including “a photo of” and other abstract terms) to keep their shape, or only select the other salient objects and hold those fixed. In general, since abstract words are often used for message passing and have attention patterns that are correlated with the layout of the scene, we prefer not to guide their layouts to maximize compositionality.

Mitigating appearance-layout entanglement. When words or concepts span multiple tokens, we can mean-pooling attention maps across these tokens before processing them, though do not find this to improve results. We also find that corrupting target shapes with Gaussian noise helps mitigate this effect, providing some evidence for this hypothesis.

Moving objects. We use $w_0 \in [0.5, 2], w_1 \in [0.03, 0.3], w_2 \in [0.5, 5]$ in Eqn. 3.9. Alternatively, we can express o_k's new location through its `centroid`, adding a term to keep `size` fixed:

$$\begin{aligned} g = w_0 & \overbrace{\frac{1}{|O|-1}\sum_{o\neq o_k}\frac{1}{|\mathcal{A}|}\sum_{i=0}^{|\mathcal{A}|}\|\texttt{shape}_{i,t,\text{orig}}(o)-\texttt{shape}_{i,t}(o)\|_1}^{\text{Fix all other object shapes}} \\ + w_1 & \overbrace{\frac{1}{|O|}\sum_{o\in O}\|\texttt{appearance}_{t,\text{orig}}(o)-\texttt{appearance}_{t}(o)\|_1}^{\text{Fix all object appearances}} \\ + w_2 & \overbrace{\frac{1}{|\mathcal{A}|}\sum_{i=0}^{|\mathcal{A}|}\|\texttt{size}_{i,t,\text{orig}}(o_k)-\texttt{size}_{i,t}(o_k)\|_1}^{\text{Fix } o_k\text{'s size}} \\ + w_3 & \underbrace{\frac{1}{|\mathcal{A}|}\sum_{i=0}^{|\mathcal{A}|}\|\texttt{target\_centroid}-\texttt{centroid}_{i,t}(o_k)\|_1}_{\text{Change } o_k\text{'s position}} \end{aligned} \tag{3.17}$$

Where $\texttt{target\_centroid}$ can be computed as a shfited version of the timestep-and-attention-specific $\texttt{centroid}_{\text{orig}}$ if desired, or selected to be an absolute value on the canvas (repeated across all timesteps). We generally use weights $w_0 \in [0.5, 2], w_1 \in [0.03, 0.3], w_2 \in [0.5, 2], w_3 \in [1, 3]$.

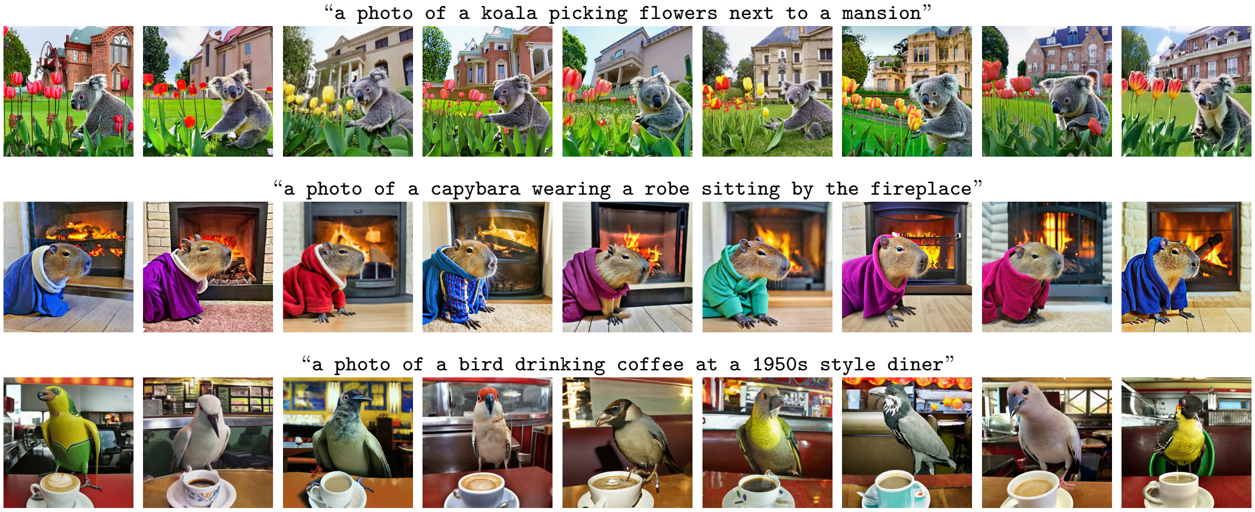
Resizing objects. We can follow Eqn. 3.9 to resize objects as well, by setting $\mathcal{T}$ to upsample or downsample the original mask. We can similarly use Eqn. 3.17, omitting the final term and setting the target $\texttt{size}$ to a desired value, either computed as a function of $\texttt{size}_{\text{orig}}(o_k)$ or provided as an absolute proportion of pixels on the canvas that the object should cover. We use the same weight range for all weights except we set $w_2 \in [1, 3], w_3 = 0$ for Eqn. 3.17.

Sampling new appearances. We set $w_0 \in [0.1, 1]$ in Eqn. 3.10. Generally, higher values lead to extremely precise layout preservation at the expense of diversity in appearance.

Sampling new layouts. Just as we can find new appearances for a scene of a given layout, we can perform the opposite operation, finding new layouts for scenes where objects have a given appearance:

$$g = w_0 \overbrace{\frac{1}{|O|}\sum_{o\in O}\|\texttt{appearance}_{t,\text{orig}}(o)-\texttt{appearance}_{t}(o)\|_1}^{\text{Fix all appearances}} \tag{3.18}$$

We almost always use $w_0 \in [0.05, 0.25]$.



“a photo of a koala picking flowers next to a mansion”

“a photo of a capybara wearing a robe sitting by the fireplace”

“a photo of a bird drinking coffee at a 1950s style diner”

Original (b) Edited

Figure 3.13: **Creating new appearances for scenes.** Non-cherry-picked results sampling different “styles” of appearances given the same layout, using Eqn. 3.10. We use $w_0 = 0.7$, 0.3, and 0.3 respectively for each result, to preserve greater structure in the background of the first picture.

Collaging objects in-place. Eqn. 3.12 can be easily generalized to more than one object per image (adding another sum across all objects) or to the case where prompts vary between images (mapping from k_j to the corresponding indices in the new image). We set $w_0 \in [0.5, 1], w_1 \in [0.05, 0.3]$.

Collaging objects with a new layout. As shown in Fig. 3.6f, we can also collage objects into a new layout specified by a target image $J + 1$, in addition to the J images specifying object appearance:

$$
\begin{aligned}
g = w_0 & \overbrace{\frac{1}{|\mathcal{A}|} \sum_{i=0}^{|\mathcal{A}|} \|\texttt{shape}_{i,t,J+1}(o_{k_{J+1}}) - \texttt{shape}_{i,t}(o_k)\|_1}^{\text{Copy all object shapes}} \\
& + w_1 \overbrace{\frac{1}{J} \sum_j \|\texttt{appearance}_{t,j}(o_{k_j}) - \texttt{appearance}_t(o_k)\|_1}^{\text{Copy each object's appearance}}
\end{aligned}
\tag{3.19}
$$

As in Eqn. 3.12, we set $w_0 \in [0.5, 1], w_1 \in [0.05, 0.3]$.

Transferring object appearances to new layouts. Nothing requires the indices (or in fact, the objects those indices refer to) to be the same in the image being generated and the

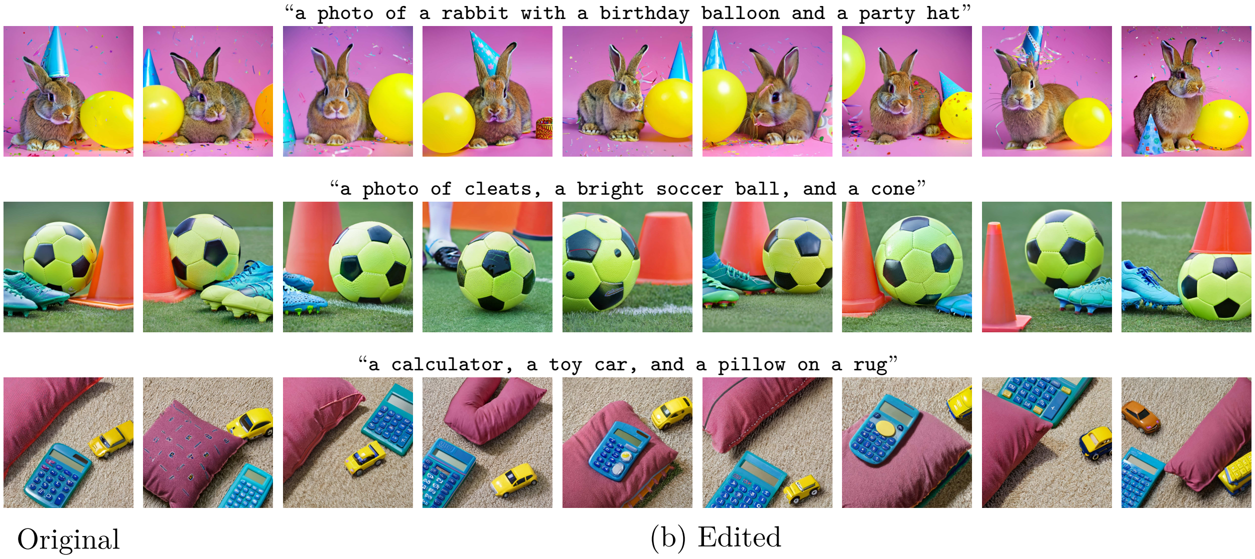

"a photo of a rabbit with a birthday balloon and a party hat"
"a photo of cleats, a bright soccer ball, and a cone"
"a calculator, a toy car, and a pillow on a rug"
Original
(b) Edited

Figure 3.14: **Creating new layouts for scenes.** Non-cherry-picked results sampling new layouts for the same scenes, using Eqn. 3.18. We use $w_0 = 0.07$, 0.07, and 0.2 respectively.

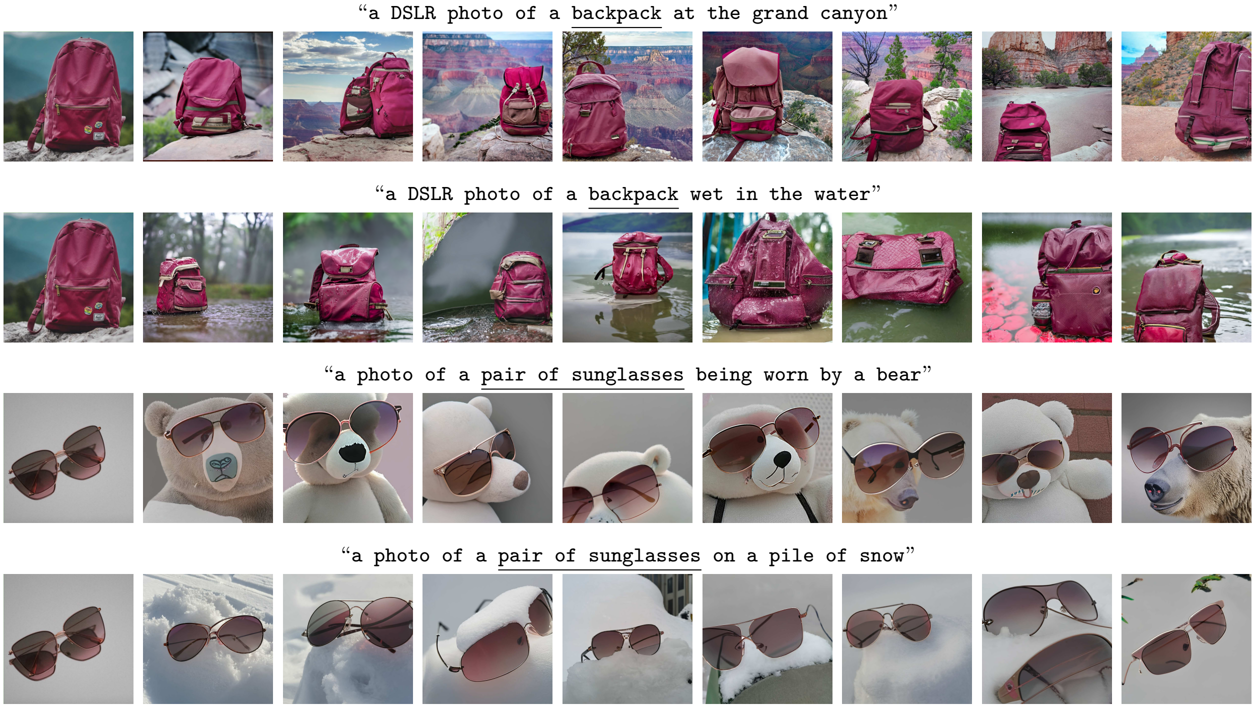
original image being used as a source, as long as there is a mapping specified between the indices in the old and new images which should correspond. Call this mapping m. We can then take the appearance of an object o_k in a source image and transfer it to an image with any new prompt as follows, as specified in Eqn. 3.13 (with typical weights $w_0 \in [0.01, 0.1]$):

$$g = w_0 \overbrace{\|\texttt{appearance}_{t,\text{orig}}(o_k) - \texttt{appearance}_t\,(m(o_k))\,\|_1}^{\text{Copy object appearance}} \tag{3.20}$$

Merging layout and appearance. We use $w_0 \in [1, 2]$ and $w_1 \in [0.1, 0.3]$ in Eqn. 3.11.

Editing with real images. Importantly, our method is not limited to editing generated images to whose internals it has access by definition. We find that we can also meaningfully guide generation using the attention and activations extracted from a set of forward-process denoisings of a real image (given a caption) to "approximate" the reverse process, despite any mismatch in distributions one might imagine. Concretely, we generate T corrupted versions of a real image x, $\{\alpha_t x + \sigma_t \epsilon_t\}_1^T$, where $\epsilon_t \sim \mathcal{N}(0, 1)$. We then extract the attention $\mathcal{A}_t$ and activations Ψ_t from the denoising network at each of these timesteps in parallel and concatenate them into a length-T sequence. We treat this sequence identically to a sequence of T internals given by subsequent sampling steps, and can thus transfer the appearance of objects from real images, output images that look like real images with moved or resized objects, and so on.

In Fig. 3.7, the prompts we use to transfer appearance are "A photo of a Chow Chow..." and "A DSLR photo of a teapot...". While our method works on less specific descriptions as


“a DSLR photo of a backpack at the grand canyon”

“a DSLR photo of a backpack wet in the water”

“a photo of a pair of sunglasses being worn by a bear”

“a photo of a pair of sunglasses on a pile of snow”

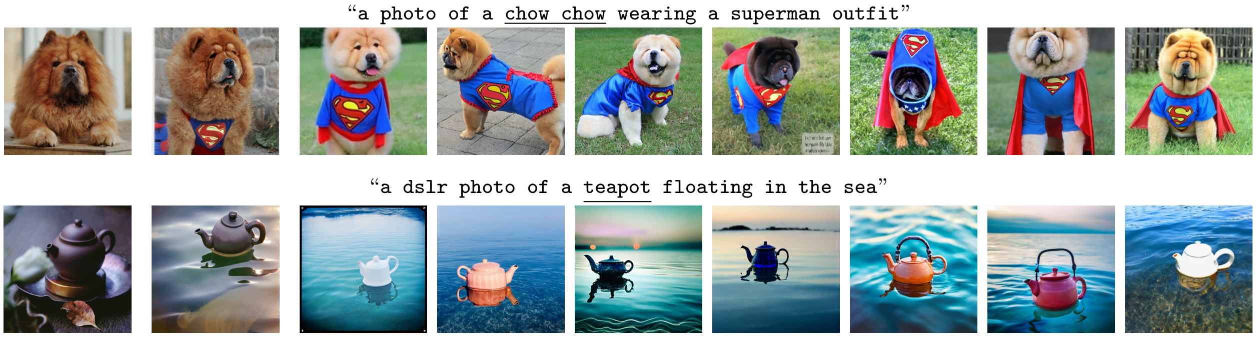
Original

(b) Object in new contexts

Figure 3.15: **Appearance transfer from real images.** Non-cherry picked results sampling new images with a given object’s appearance specified by a real images, as in Eqn. 3.13. We use $w_0 = 0.15$.

well, it is not as reliable when object appearance is more out-of-distribution. For context, we show unguided samples under the prompts from Fig. 3.7 in Fig. 3.16, which still deviate significantly from the desired appearance, showing the efficacy of our approach. A weakness of our simple approach is that it has no constraints on the shape of the generated objects, which we leave to future work.

Weight selection heuristics. We find weights that work well to remain more or less consistent across different images given an edit, but ideal weights do vary somewhat (within predictable ranges) between different combinations of terms. Our heuristics for weight selection per term are: the more weights there are, the higher per-term weights can be without causing artifacts (and indeed, need to be, to provide ample contribution to the final result); `appearance` terms should have weights 1 or 2 orders of magnitude lower than layout terms; layout summary statistics (`centroid` and `size`) should have slightly lower weights than terms on the per-pixel `shape`; total weight of terms should not add up to more than ~ 5 to avoid artifacts.


“a photo of a chow chow wearing a superman outfit”
“a dslr photo of a teapot floating in the sea”

Original Ours (b) Random samples without self-guidance

Figure 3.16: **Ablating appearance transfer from real images.** To verify the efficacy of our approach, we compare our results from Fig. 3.7 in the paper to random samples from the same prompt without apperance transfer. We can see that appearance of objects varies significantly without self-guidance.

3.6.3 Additional results

We show further non-cherry-picked results for the edits we show in the main paper. Our general protocol consists of selecting an interesting prompt manually, verifying that our model creates compelling samples aligning with this prompt without self-guidance, beginning with the typical weights we use for an edit, and trying around 3-5 other weight configurations to find the one that works best for the prompt – in most cases, this is the starting set of weights. Then, we use the first 8 images we generate, without further filtering. We generate all results with different seeds to showcase the strength of guidance even without shared DDPM noise. We show more results for moving (Fig. 3.11) and resizing (Fig. 3.12) objects, sampling new appearances for given layouts (Fig. 3.13) as well as new layouts for a given set of objects (Fig. 3.14), and transferring the appearance of real objects into new contexts (Fig. 3.15). We also include an ablation on hyperparameter values (Fig. 3.17) as well as preliminary results of an implementation of self-guidance on an open-source diffusion model in Fig. 3.18.

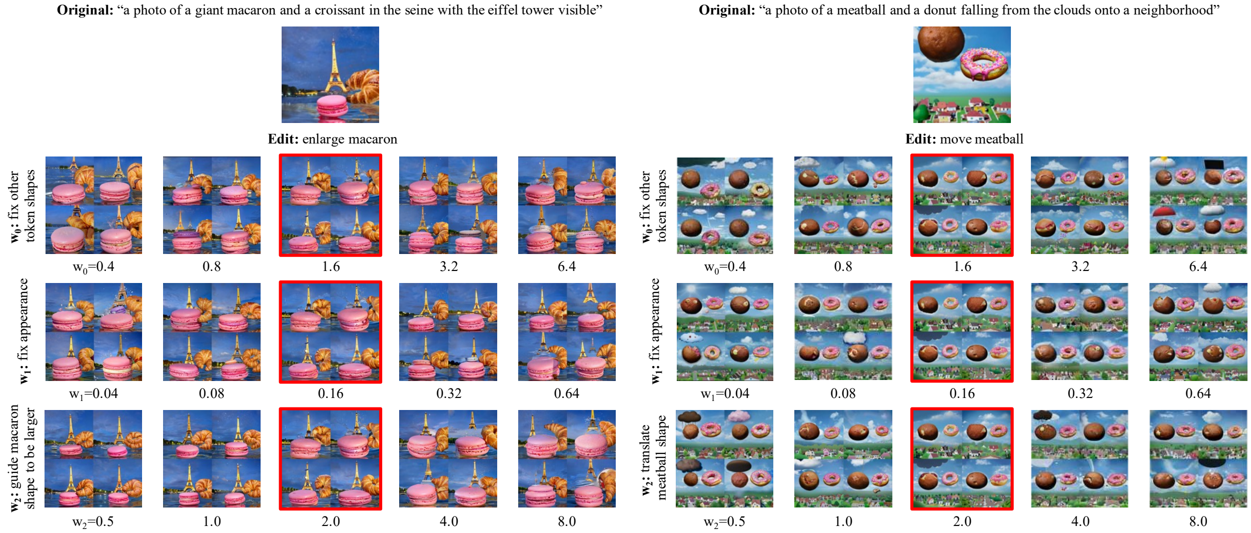
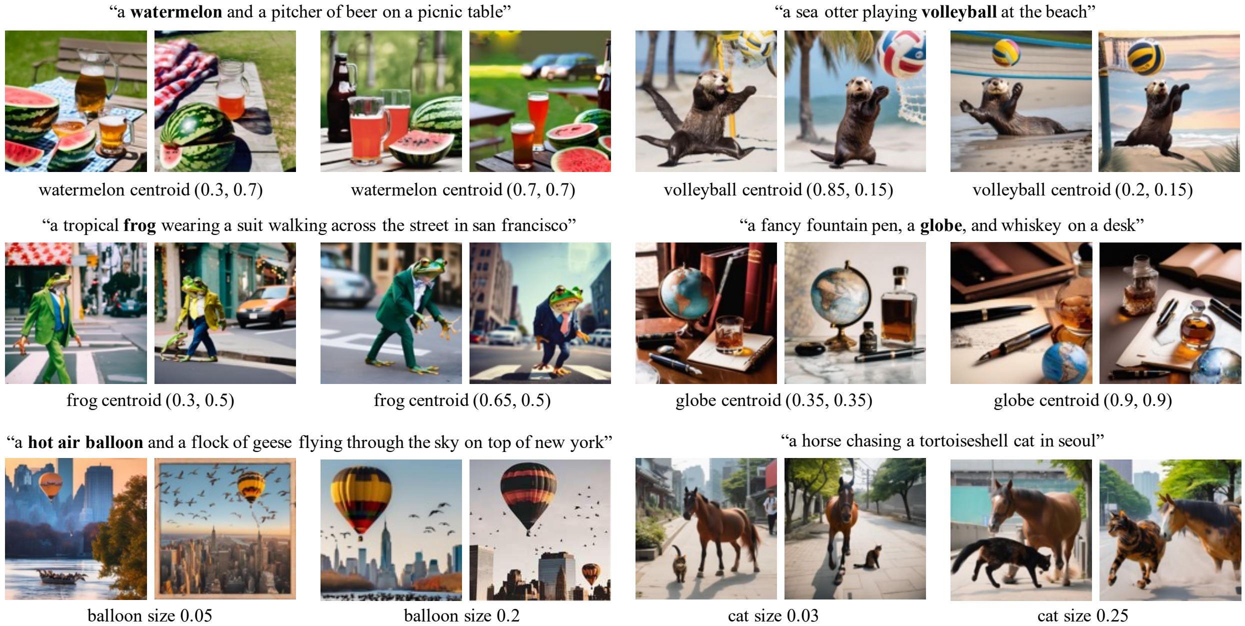

Original: "a photo of a giant macaron and a croissant in the seine with the eiffel tower visible"
Edit: enlarge macaron
w$_0$: fix other token shapes
w$_0$=0.4
0.8
1.6
3.2
6.4
w$_1$: fix appearance
w$_1$=0.04
0.08
0.16
0.32
0.64
w$_2$: guide macaron shape to be larger
w$_2$=0.5
1.0
2.0
4.0
8.0
Original: "a photo of a meatball and a donut falling from the clouds onto a neighborhood"
Edit: move meatball
w$_0$: fix other token shapes
w$_0$=0.4
0.8
1.6
3.2
6.4
w$_1$: fix appearance
w$_1$=0.04
0.08
0.16
0.32
0.64
w$_2$: translate meatball shape
w$_2$=0.5
1.0
2.0
4.0
8.0

Figure 3.17: **Hyperparameter sweeps.** We show results for two edits (moving and resizing objects) using Eqn. 4 in the Supp. Mat., for different values of weights on the three edit terms, holding the other terms to the value in the middle column. Please zoom in to view in more detail. Reasonable values in the middle columns (within the expected range) lead to overall successful image manipulation. Very large hyperparameter values cause visual artifacts to appear (by moving sampling off-manifold) while still tending to perform the edit successfully, while extremely small values often fail to conduct the edit, inducing artifacts resulting from a "half-executed" manipulation.



"a **watermelon** and a pitcher of beer on a picnic table"

watermelon centroid (0.3, 0.7)

watermelon centroid (0.7, 0.7)

"a sea otter playing **volleyball** at the beach"

volleyball centroid (0.85, 0.15)

volleyball centroid (0.2, 0.15)

"a tropical **frog** wearing a suit walking across the street in san francisco"

frog centroid (0.3, 0.5)

frog centroid (0.65, 0.5)

"a fancy fountain pen, a **globe**, and whiskey on a desk"

globe centroid (0.35, 0.35)

globe centroid (0.9, 0.9)

"a **hot air balloon** and a flock of geese flying through the sky on top of new york"

balloon size 0.05

balloon size 0.2

"a horse chasing a tortoiseshell cat in seoul"

cat size 0.03

cat size 0.25

Figure 3.18: **Self-guidance on Stable Diffusion XL.** To highlight the generality of our approach, we demonstrate preliminary results for controllable generation on a popular latent-space text-to-image diffusion model, using 100 DDPM steps (applying self-guidance from step 10 to step 90). We get best results only guiding attention in the second decoder block of the denoiser model.

Chapter 4

An Architecture for Disentangled 3D Scenes

Inspired by the richness of the distribution captured by large text-to-image models (made apparent in the previous chapter), we ask whether these models also capture information about our world in three dimensions. We introduce a method to generate 3D scenes that are disentangled into their component objects. This disentanglement is unsupervised, relying only on the knowledge of the pretrained model. Our key insight is that objects can be discovered by finding parts of a 3D scene that, when rearranged spatially, still produce valid configurations of the same scene. Concretely, our method jointly optimizes multiple NeRFs from scratch—each representing its own object—along with a *set of layouts* that composite these objects into scenes. We then encourage these composited scenes to be in-distribution according to the image generator. We show that despite its simplicity, our approach successfully generates 3D scenes decomposed into individual objects, enabling new capabilities in text-to-3D content creation.

4.1 Introduction

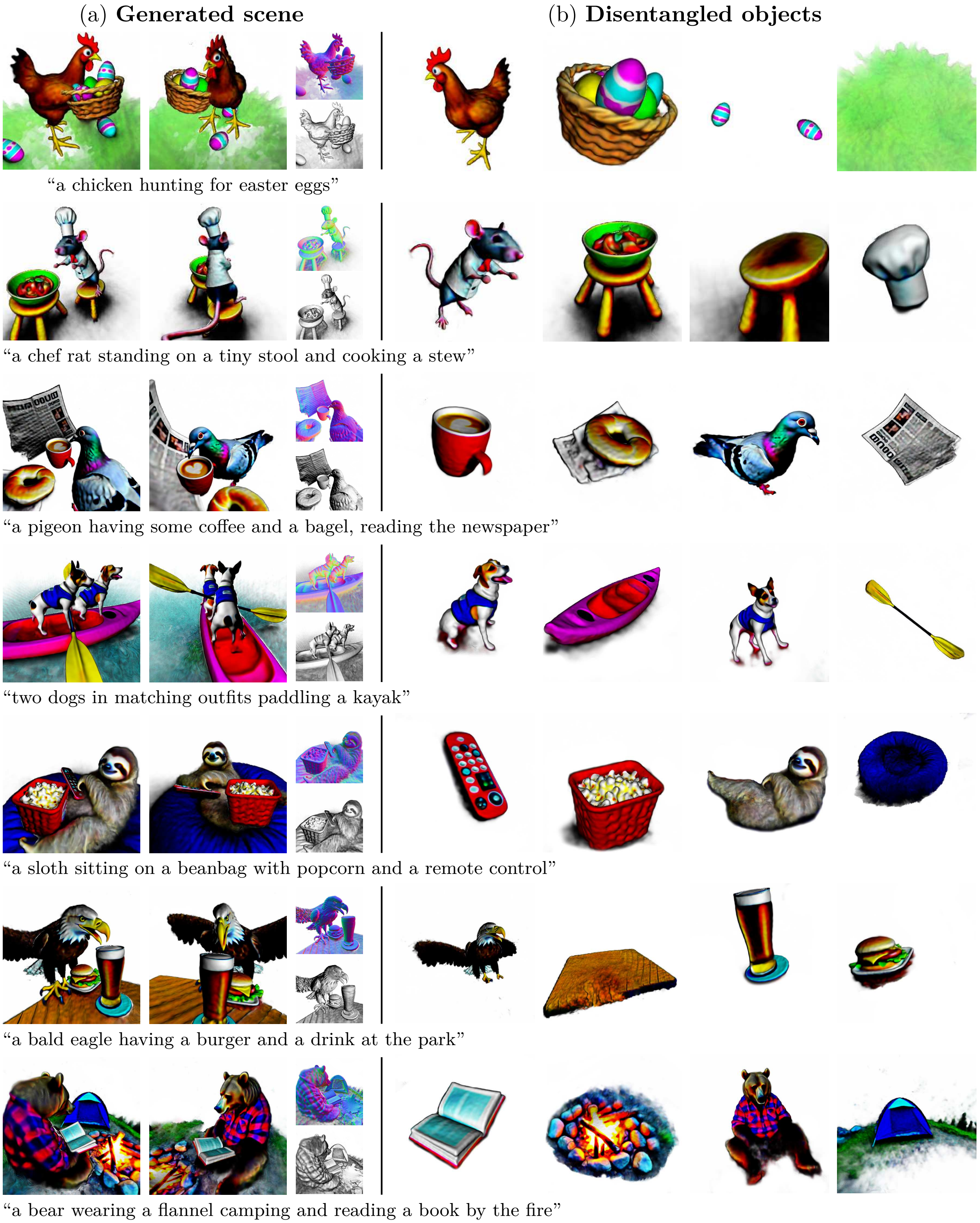
A remarkable ability of many seeing organisms is object individuation [128], the ability to discern separate objects from light projected onto the retina [183]. Indeed, from a very young age, humans and other creatures are able to organize the physical world they perceive into the three-dimensional entities that comprise it [167, 184, 58]. The analogous task of object discovery has captured the attention of the artificial intelligence community from its very inception [140, 118], since agents that can autonomously parse 3D scenes into their component objects are better able to navigate and interact with their surroundings.

Fifty years later, generative models of images are advancing at a frenzied pace [114, 136, 152, 197, 22]. While these models can generate high-quality samples, their internal workings are hard to interpret, and they do not explicitly represent the distinct 3D entities that make up the images they create. Nevertheless, the priors learned by these models have proven incredibly useful across various tasks involving 3D reasoning [50, 78, 95, 99, 185], suggesting that they may indeed be capable of decomposing generated content into the underlying 3D objects depicted.

One particularly exciting application of these text-to-image networks is 3D generation, leveraging the rich distribution learned by a diffusion model to optimize a 3D representation, *e.g.* a neural radiance field (NeRF, Mildenhall et al., 2020), such that rendered views resemble samples from the prior. This technique allows for text-to-3D generation without any 3D supervision [130, 179], but most results focus on simple prompts depicting just one or two isolated objects [93, 182].

Our method builds on this work to generate complex scenes that are automatically disentangled into the objects they contain. To do so, we instantiate and render *multiple NeRFs* for a given scene instead of just one, encouraging the model to use each NeRF to represent a separate 3D entity. At the crux of our approach is an intuitive definition of objects as parts of a scene that can be manipulated independently of others while keeping the scene "well-formed" [14]. We implement this by learning a set of different layouts—3D affine transformations of every NeRF—which must yield composited scenes that render into in-distribution 2D images given a text prompt [130].

We find that this lightweight inductive bias, which we term *layout learning*, results in surprisingly effective object disentanglement in generated 3D scenes (Figure 4.1), enabling object-level scene manipulation in the text-to-3D pipeline. We demonstrate the utility of layout learning on several tasks, such as building a scene around a 3D asset of interest, sampling different plausible arrangements for a given set of assets, and even parsing a provided NeRF into the objects it contains, all without any supervision beyond just a text prompt. We further quantitatively verify that, despite requiring no auxiliary models or per-example human annotation, the object-level decomposition that emerges through layout learning is meaningful and outperforms baselines.



(a) **Generated scene**

(b) **Disentangled objects**

“a chicken hunting for easter eggs”

“a chef rat standing on a tiny stool and cooking a stew”

“a pigeon having some coffee and a bagel, reading the newspaper”

“two dogs in matching outfits paddling a kayak”

“a sloth sitting on a beanbag with popcorn and a remote control”

“a bald eagle having a burger and a drink at the park”

“a bear wearing a flannel camping and reading a book by the fire”

Figure 4.1: **Layout learning generates disentangled 3D scenes** given a text prompt and a pretrained text-to-image diffusion model. We learn an entire 3D scene (left, shown from two views along with surface normals and a textureless render) that is composed of multiple NeRFs (right) representing different objects and arranged according to a learned layout.

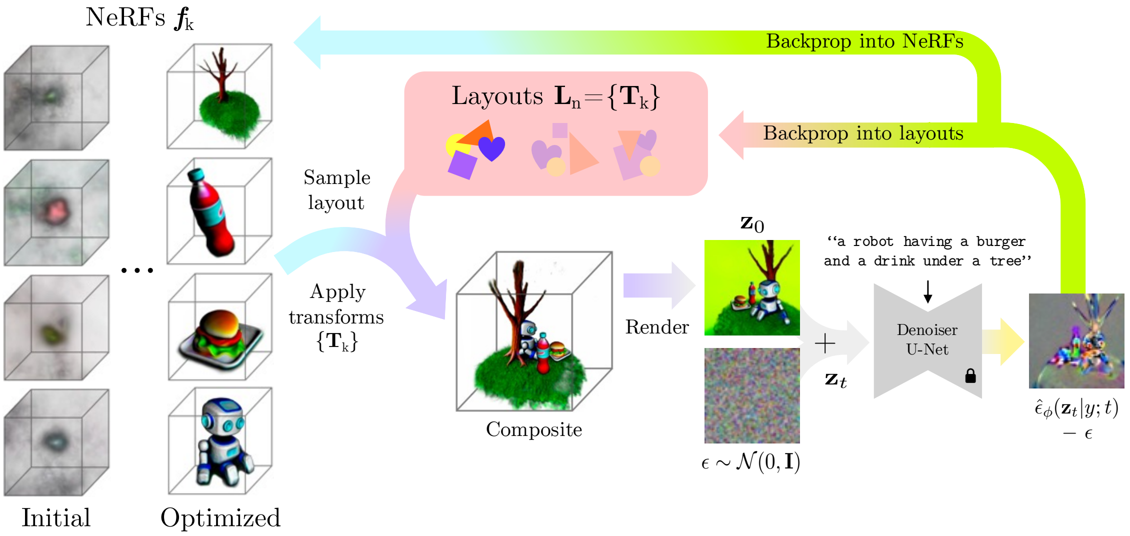

NeRFs f_k
Backprop into NeRFs
Layouts $\mathbf{L}_n = \{\mathbf{T}_k\}$
Backprop into layouts
Sample layout
...
Apply transforms $\{\mathbf{T}_k\}$
Composite
Render
$\mathbf{z}_0$
“a robot having a burger and a drink under a tree”
Denoiser U-Net
$\mathbf{z}_t$
$\epsilon \sim \mathcal{N}(0, \mathbf{I})$
$\hat{\epsilon}_\phi(\mathbf{z}_t|y;t) - \epsilon$
Initial
Optimized

Figure 4.2: **Method.** Layout learning works by optimizing K NeRFs f_k and learning N different layouts $\mathbf{L}_n$ for them, each consisting of per-NeRF affine transforms $\mathbf{T}_k$. Every iteration, a random layout is sampled and used to transform all NeRFs into a shared coordinate space. The resultant volume is rendered and optimized with score distillation sampling [130] as well as per-NeRF regularizations to prevent degenerate decompositions and geometries [7]. This simple structure causes object disentanglement to emerge in generated 3D scenes.

Our key contributions are as follows:

- We introduce a simple, tractable definition of objects as portions of a scene that can be manipulated independently of each other and still produce valid scenes.
- We incorporate this notion into the architecture of a neural network, enabling the compositional generation of 3D scenes by optimizing a set of NeRFs as well as a set of layouts for these NeRFs.
- We apply layout learning to a range of novel 3D scene generation and editing tasks, demonstrating its ability to disentangle complex data despite requiring no object labels, bounding boxes, fine-tuning, external models, or any other form of additional supervision.

4.2 Background

4.2.1 Neural 3D representations

To output three-dimensional scenes, we must use an architecture capable of modeling 3D data, such as a neural radiance field (NeRF, Mildenhall et al., 2020). We build on MLP-based

NeRFs [8], that represent a volume using an MLP f that maps from a point in 3D space $\boldsymbol{\mu}$ to a density τ and albedo $\boldsymbol{\rho}$:

$$(\tau, \boldsymbol{\rho}) = f(\boldsymbol{\mu}; \theta).$$

We can differentiably render this volume by casting a ray $\mathbf{r}$ into the scene, and then alpha-compositing the densities and colors at sampled points along the ray to produce a color and accumulated alpha value. For 3D reconstruction, we would optimize the colors for the rendered rays to match a known pixel value at an observed image and camera pose, but for 3D generation we sample a random camera pose, render the corresponding rays, and score the resulting image using a generative model.

4.2.2 Text-to-3D using 2D diffusion models

Our work builds on text-to-3D generation using 2D diffusion priors [130]. These methods turn a diffusion model into a loss function that can be used to optimize the parameters of a 3D representation. Given an initially random set of parameters θ, at each iteration we randomly sample a camera c and render the 3D model to get an image $x = g(\theta, c)$. We can then score the quality of this rendered image given some conditioning text y by evaluating the score function of a noised version of the image $z_t = \alpha_t x + \sigma_t \epsilon$ using the pretrained diffusion model $\hat{\epsilon}(z_t; y, t)$. We update the parameters of the 3D representation using score distillation:

$$\nabla_\theta \mathcal{L}_{\text{SDS}}(\theta) = \mathbb{E}_{t,\epsilon,c}\left[w(t)(\hat{\epsilon}(z_t; y, t) - \epsilon)\frac{\partial x}{\partial \theta} \right] \tag{4.1}$$

where $w(t)$ is a noise-level dependent weighting.

SDS and related methods enable the use of rich 2D priors obtained from large text-image datasets to inform the structure of 3D representations. However, they often require careful tuning of initialization and hyperparameters to yield high quality 3D models, and past work has optimized these towards object generation. The NeRF is initialized with a Gaussian blob of density at the origin, biasing the optimization process to favor an object at the center instead of placing density in a skybox-like environment in the periphery of the 3D representation. Additionally, bounding spheres are used to prevent creation of density in the background. The resulting 3D models can produce high-quality individual objects, but often fail to generate interesting scenes, and the resulting 3D models are a single representation that cannot be easily split apart into constituent entities.

4.3 Method

To bridge the gap from monolithic 3D representations to scenes with multiple objects, we introduce a more expressive 3D representation. Here, we learn multiple NeRFs along with a set of layouts, *i.e.* valid ways to arrange these NeRFs in 3D space. We transform the NeRFs according to these layouts and composite them, training them to form high-quality scenes as evaluated by the SDS loss with a text-to-image prior. This structure causes each individual NeRF to represent a different object while ensuring that the composite NeRF represents a high-quality scene. See Figure 4.2 for an overview of our approach.

4.3.1 Compositing multiple volumes

We begin by considering perhaps the most naïve approach to generating 3D scenes disentangled into separate entities. We simply declare K NeRFs $\{f_k\}$—each one intended to house its own object—and jointly accumulate densities from all NeRFs along a ray, proceeding with training as normal by rendering the composite volume. This can be seen as an analogy to set-latent representations [98, 65, 66, 64], which have been widely explored in other contexts. In this case, rather than arriving at the final albedo $\boldsymbol{\rho}$ and density τ of a point $\boldsymbol{\mu}$ by querying one 3D representation, we query K such representations, obtaining a set $\{\boldsymbol{\rho}_k, \tau_k\}_{k=1}^K$. The final density at $\boldsymbol{\mu}$ is then $\tau' = \sum \tau_k$ and the final albedo is the density-weighted average $\boldsymbol{\rho}' = \sum \frac{\tau_k}{\tau'} \boldsymbol{\rho}_k$.

This formulation provides several potential benefits. First, it may be easier to optimize this representation to generate a larger set of objects, since there are K distinct 3D Gaussian density spheres to deform at initialization, not just one. Second, many representations implicitly contain a local smoothness bias [172] which is helpful for generating objects but not spatially discontinuous scenes. Thus, our representation might be inclined toward allocating each representation toward a spatially smooth entity, *i.e.* an object.

However, just as unregularized sets of latents are often highly uninterpretable, simply spawning K instances of a NeRF does not produce meaningful decompositions. In practice, we find each NeRF often represents a random point-cloud-like subset of 3D space (Fig. 4.3).

To produce scenes with disentangled objects, we need a method to encourage each 3D instance to represent a coherent object, not just a different part of 3D space.

4.3.2 Layout learning

We are inspired by other unsupervised definitions of objects that operate by imposing a simple inductive bias or regularization in the structure of a model's latent space, *e.g.* query-axis softmax attention [98], spatial ellipsoid feature maps [33], and diagonal Hessian matrices [127]. In particular, [116] learn a 3D-aware GAN that composites multiple NeRF volumes in the forward pass, where the latent code contains a random affine transform for each NeRF's output. Through this structure, each NeRF learns to associate itself with a different object, facilitating the kind of disentanglement we are after. However, their approach relies on pre-specified independent distributions of each object's location, pose, and size, preventing scaling beyond narrow datasets of images with one or two objects and minimal variation in layout.

In our setting, not only does the desired output comprise numerous open-vocabulary, arbitrary objects, but these objects *must be arranged in a particular way* for the resultant scene to be valid or "well-formed" [15]. Why not simply learn this arrangement?

To do this, we equip each individual NeRF f_k with its own learnable affine transform $\mathbf{T}_k$, and denote the set of transforms across all volumes a layout $\mathbf{L} \equiv \{\mathbf{T}_k\}_{k=1}^{K}$. Each $\mathbf{T}_k$ has a rotation $\mathbf{R}_k \in \mathbb{R}^{3\times3}$ (in practice expressed via a quaternion $\mathbf{q} \in \mathbb{R}^4$ for ease of optimization), translation $\mathbf{t}_k \in \mathbb{R}^3$, and scale $s_k \in \mathbb{R}$. We apply this affine transform to the camera-to-world rays $\mathbf{r}$ before sampling the points used to query f_k. This implementation is simple, makes no assumptions about the underlying form of f, and updates parameters with standard backpropagation, as sampling and embedding points along the ray is fully differentiable [92]. Concretely, a ray $\mathbf{r}$ with origin $\mathbf{o}$ and direction $\mathbf{d}$ is transformed into an instance-specific ray $\mathbf{r}_k$ via the following transformations:

$$\mathbf{o}_k = s_k \left(\mathbf{R}_k \mathbf{o} - \mathbf{t}_k\right) \tag{4.2}$$

$$\mathbf{d}_k = s_k \mathbf{R}_k \mathbf{d} \tag{4.3}$$

$$\mathbf{r}_k(t) = \mathbf{o}_k + t\mathbf{d}_k \tag{4.4}$$

Though we input a different $H \times W$ grid of rays to each f_k, we composite their outputs as if they all sit in the same coordinate space—for example, the final density at $\boldsymbol{\mu} = \mathbf{r}(t)$ is the sum of densities output by every f_k at $\boldsymbol{\mu}_k = \mathbf{r}_k(t)$.

Compared to the naïve formulation that instantiates K models with identical initial densities, learning the size, orientation, and position of each model makes it easier to place density in different parts of 3D space. In addition, the inherent stochasticity of optimization may further dissuade degenerate solutions.

While introducing layout learning significantly increases the quality of object disentanglement (Tbl. 4.3b), the model is still able to adjoin and utilize individual NeRFs in undesirable ways. For example, it can still place object parts next to each other in the same way as K NeRFs without layout learning.

Learning multiple layouts. We return to our statement that objects must be "arranged in a particular way" to form scenes that render to in-distribution images. While we already enable this with layout learning in its current form, we are not taking advantage of one key fact: there are many "particular ways" to arrange a set of objects, each of which gives an equally valid composition. Rather than only learning one layout, we instead learn a distribution over layouts $P(\mathbf{L})$ or a set of N randomly initialized layouts $\{\mathbf{L}_n\}_{n=1}^{N}$. We opt for the latter, and sample one of the N layouts from the set at each training step to yield transformed rays $\mathbf{r}_k$.

With this in place, we have arrived at our final definition of objectness (Figure 4.2): **objects are parts of a scene that can be arranged in different ways to form valid compositions.** We have "parts" by incorporating multiple volumes, and "arranging in different ways" through multiple-layout learning. This simple approach is easy to implement (Fig. 4.9), adds very few parameters ($8NK$ to be exact), requires no fine-tuning or manual annotation, and is agnostic to choices of text-to-image and 3D model. In Section 4.4, we verify that layout learning enables the generation and disentanglement of complex 3D scenes.

Regularization. We build on Mip-NeRF 360 [7] as our 3D backbone, inheriting their orientation, distortion, and accumulation losses to improve visual quality of renderings and minimize artifacts. However, rather than computing these losses on the final composited scene, we apply them on a per-NeRF basis. Importantly, we add a loss penalizing degenerate empty NeRFs by regularizing the soft-binarized version of each NeRF's accumulated density, $\boldsymbol{\alpha}_{\text{bin}}$, to occupy at least 10% of the canvas:

$$\mathcal{L}_{\text{empty}} = \max\left(0.1 - \bar{\boldsymbol{\alpha}}_{\text{bin}}, 0\right) \tag{4.5}$$

We initialize parameters $s \sim \mathcal{N}(1, 0.3)$, $\mathbf{t}^{(i)} \sim \mathcal{N}(0, 0.3)$, and $\mathbf{q}^{(i)} \sim \mathcal{N}(\mu_i, 0.1)$ where μ_i is 1 for the last element and 0 for all others. We use a 10× higher learning rate to train layout parameters. See Appendix 4.7.1 for more details.

4.4 Experiments

We examine the ability of layout learning to generate and disentangle 3D scenes across a wide range of text prompts. We first verify our method's effectiveness through an ablation study and comparison to baselines, and then demonstrate various applications enabled by layout learning.

4.4.1 Qualitative evaluation

In Figure 4.1, we demonstrate several examples of our full system with layout learning. In each scene, we find that the composited 3D generation is high-quality and matches the text prompt, while the individual NeRFs learn to correspond to objects within the scene. Interestingly, since our approach does not directly rely on the input prompt, we can disentangle entities not mentioned in the text, such as a basket filled with easter eggs, a chef's hat, and a picnic table.

4.4.2 Quantitative evaluation

Measuring the quality of text-to-3D generation remains an open problem due to a lack of ground truth data—there is no "true" scene corresponding to a given prompt. Similarly, there is no true disentanglement for a certain text description. Following [122, 68, 130], we attempt to capture both of these aspects using scores from a pretrained CLIP model [134, 88]. Specifically, we create a diverse list of 30 prompts, each containing 3 objects, and optimize a model with $K = 3$ NeRFs on each prompt. We compute the 3×3 matrix of CLIP scores (100× cosine similarity) for each NeRF with descriptions "a DSLR photo of [object 1/2/3]", finding the optimal NeRF-to-object matching and reporting the average score across all 3 objects.

We also run SDS on the $30 \times 3 = 90$ per-object prompts individually and compute scores, representing a maximum attainable CLIP score under perfect disentanglement (we equalize parameter counts across all models for fairness). As a low-water mark, we compute scores between per-object NeRFs and a random other prompt from the pool of 90.

The results in Table 4.3b show these CLIP scores, computed both on textured ("Color") and textureless, geometry-only ("Geo") renders. The final variant of layout learning achieves competitive performance, only 0.1 points away from supervised per-object rendering when using the largest CLIP model as an oracle, indicating high quality of both object disentanglement and appearance. Please see Appendix 4.7.3 for a complete list of prompts and more details.

Ablation. We justify the sequence of design decisions presented in Section 4.3 by evaluating different variants of layout learning, starting from a simple collection of K NeRFs and building up to our final architecture. The simple setting leads to some non-trivial separation (Figure 4.3a) but parts of objects are randomly distributed across NeRFs—CLIP scores are significantly above random, but far below the upper bound. Adding regularization losses improve scores somewhat, but the biggest gains come from introducing layout learning and then co-learning N different arrangements, validating our approach.

4.4.3 Applications of layout learning

To highlight the utility of the disentanglement given by layout learning beyond generation, we apply it to various 3D editing tasks. First, we show further results on object disentanglement in Figure 4.4, but in a scenario where one NeRF is frozen to contain an object of interest, and the rest of the scene must be constructed around it. This object's layout parameters can also be frozen, for example, if a specific position or size is desired. We examine the more challenging setting where layout parameters must also be learned, and show results incorporating a grumpy cat and green motorbike into different contexts. Our model learns plausible transformations to incorporate provided assets into scenes, while still discovering the other objects necessary to complete the prompt.

In Figure 4.5, we visualize the different layouts learned in a single training run. The variation in discovered layouts is significant, indicating that our formulation can find various meaningful arrangements of objects in a scene. This allows users of our method to explore different permutations of the same content in the scenes they generate.

Inspired by this, and to test gradient flow into layout parameters, we also examine whether our method can be used to arrange off-the-shelf, frozen 3D assets into semantically valid configurations (Figure 4.6). Starting from random positions, sizes, and orientations, layouts are updated using signal backpropagated from the image model. This learns reasonable transformations, such as a rubber duck shrinking and moving inside a tub, and a shower head moving upwards and pointing so its stream is going into the tub.

Finally, we use layout learning to disentangle a pre-existing NeRF containing multiple entities, without any per-object supervision (Fig. 4.8). We do this by randomly initializing a new model and training it with a caption describing the target NeRF. We require the first layout $\mathbf{L}_1$ to create a scene that faithfully reconstructs the target NeRF in RGB space, allowing all other layouts to vary freely. We find that layout learning arrives at reasonable decompositions of the scenes it is tasked with reconstructing.

4.5 Related work

Object recognition and discovery. The predominant way to identify the objects present in a scene is to segment two-dimensional images using extensive manual annotation [82, 90, 178], but relying on human supervision introduces challenges and scales poorly to 3D data. As an alternative, an extensive line of work on *unsupervised* object discovery [149, 146, 120, 51, 163, 193, 108] proposes different inductive biases [97] that encourage awareness of objects in a scene. However, these approaches are largely restricted to either 2D images or constrained 3D data [195, 153], limiting their applicability to complex 3D scenes. At the same time, large text-to-image models have been shown to implicitly encode an understanding of entities in their internals [34], motivating their use for the difficult problem of explicit object disentanglement.

Compositional 3D generation. There are many benefits to generating 3D scenes separated into objects beyond just better control. For example, generating objects one at a time and compositing them manually provides no guarantees about compatibility in appearance or pose, such as "dogs in matching outfits" in Figure 4.1 or a lion holding the handlebars of a motorcycle in Figure 4.4. Previous and concurrent work explores this area, but either requires users to painstakingly annotate 3D bounding boxes and per-object labels [25, 129] or uses external supervision such as LLMs to propose objects and layouts [191, 203], significantly slowing down the generation process and hindering quality. We show that this entire process can be solved without any additional models or labels, simply using the signal provided by a pretrained image generator.

4.6 Discussion

We present layout learning, a simple method for generating disentangled 3D scenes given a text prompt. By optimizing multiple NeRFs to form valid scenes across multiple layouts, we encourage each NeRF to contain its own object. This approach requires no additional supervision or auxiliary models, yet performs quite well. By generating scenes that are decomposed into objects, we provide users of text-to-3D systems with more granular, local control over the complex creations output by a black-box neural network.

Though layout learning is surprisingly effective on a wide variety of text prompts, the problem of object disentanglement in 3D is inherently ill-posed, and our definition of objects is simple. As a result, many undesirable solutions exist that satisfy the constraints we pose.

Despite our best efforts, the compositional scenes output by our model do occasionally suffer from failures (Fig. 4.7) such as over- or under-segmentation and the "Janus problem" (where objects are depicted so that salient features appear from all views, *e.g.* an animal with a face on the back of its head) as well as other undesirable geometries. Further, though layouts are initialized with high standard deviation and trained with an increased learning rate, they occasionally converge to near-identical values, minimizing the effectivness of our method. In general, we find that failures to disentangle are accompanied by an overall decrease in visual quality.

4.7 Appendix

4.7.1 Implementation details

We use Mip-NeRF 360 as the 3D backbone [7] and Imagen [152], a 128px pixel-space diffusion model, for most experiments, rendering at 512px. To composite multiple representations, we merge the output albedos and densities at each point, taking the final albedo as a weighted average given by per-NeRF density. We apply this operation to the outputs of the proposal MLPs as well as the final RGB-outputting NeRFs. We use $\lambda_{\text{dist}} = 0.001, \lambda_{\text{acc}} = 0.01, \lambda_{\text{ori}} = 0.01$ as well as $\lambda_{\text{empty}} = 0.05$. The empty loss examines the mean of the per-pixel accumulated density along rays in a rendered view, $\boldsymbol{\alpha}$, for each NeRF. It penalizes these mean $\bar{\boldsymbol{\alpha}}$ values if they are under a certain fraction of the image canvas (we use 10%). For more robustness to noise, we pass $\boldsymbol{\alpha}$ through a scaled sigmoid to binarize it (Fig. 4.10), yielding the $\bar{\boldsymbol{\alpha}}_{\text{bin}}$ used in Eq. 4.5. We sample camera azimuth in $[0°, 360°]$ and elevation in $[-90°, 0°]$ except in rare cases where we sample azimuth in a 90-degree range to minimize Janus-problem artifacts or generate indoor scenes with a diorama-like effect.

We use a classifier-free guidance strength of 200 and textureless shading probability of 0.1 for SDS [130], disabling view-dependent prompting as it does not aid in the generation of compositional scenes (Table 4.3b). We otherwise inherit all other details, such as covariance annealing and random background rendering, from SDS. We optimize our model with Shampoo [44] with a batch size of 1 for 15000 steps with an annealed learning rate, starting from 10^{-9}, peaking at 10^{-4} after 3000 steps, and decaying to 10^{-6}.

Optimizing NGPs. To verify the robustness of our approach to different underlying 3D representations, we also experiment with a re-implementation of Instant NGPs [109], and find that our method generalizes to that setting. Importantly, we implement an aggressive coarse-to-fine training regime in the form of slowly unlocking grid settings at resolution higher than 64×64 only after 2000 steps. Without this constraint on the initial smoothness of geometry, the representation "optimizes too fast" and is prone to placing all density in one NGP.

4.7.2 Pseudo-code for layout learning

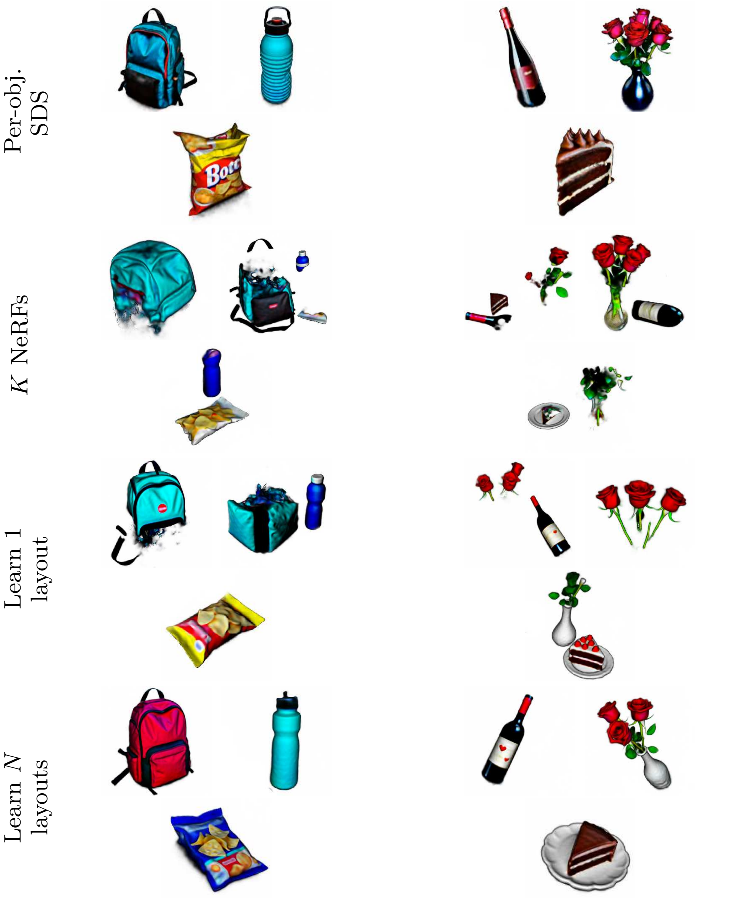
In Figs. 4.9 and 4.10, we provide NumPy-like pseudocode snippets of the core logic necessary to implement layout learning, from transforming camera rays to compositing multiple 3D volumes to regularizing them.

4.7.3 CLIP evaluation

To evaluate our approach, we use similarity scores output by a pretrained contrastive text-image model [134], which have been shown to correlate with human judgments on the quality of compositional generation [122]. However, rather than compute a retrieval-based metric such as precision or recall, we report the raw (100× upscaled, as is common practice) cosine similarities. In addition to being a more granular metric, this avoids the dependency of retrieval on the size and difficulty of the test set (typically only a few hundred text prompts).

We devise a list of 30 prompts (Fig. 4.11), each of which lists three objects, spanning a wide range of data, from animals to food to sports equipment to musical instruments. As described in Section 4.4, we then train models with $K = 3$ NeRFs and layout learning and test whether each NeRF contains a different object mentioned in the prompt. We compute CLIP scores for each NeRF with a query prompt "a DSLR photo of [A/B/C]", yielding a 3×3 score matrix.

To compute NeRF-prompt CLIP scores, we average text-image similarity across 12 uniformly sampled views, each 30 degrees apart, at -30° elevation. We then select the best NeRF-prompt assignment (using brute force, as there are only $3! = 6$ possible choices), and run this process across 3 different seeds, choosing the one with the highest mean NeRF-prompt score.

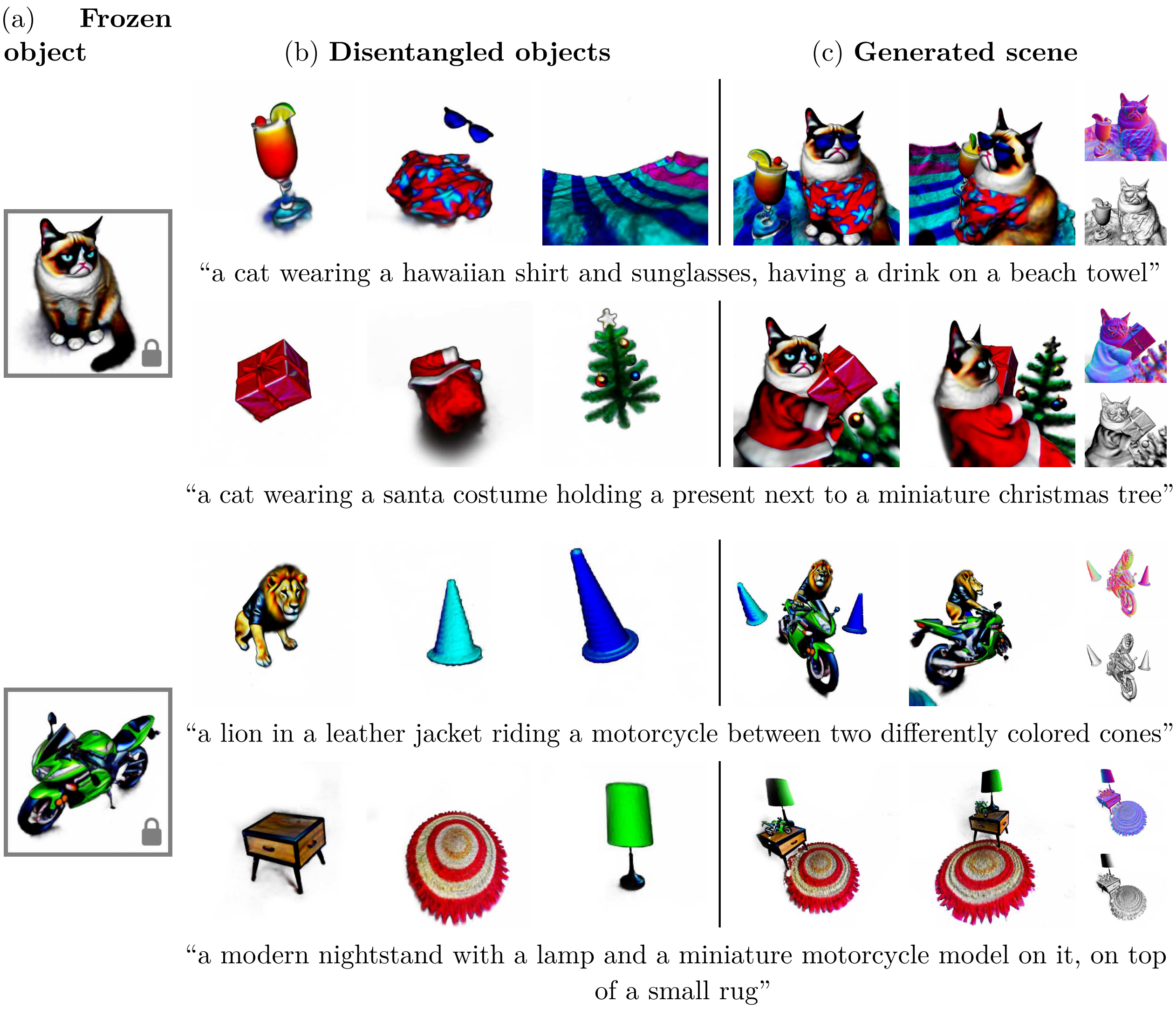


"a backpack, water bottle, and bag of chips"

"a slice of cake, vase of roses, and bottle of wine"

	Average Score ↑			
Method	CLIP B/16		CLIP L/14	
	Color	Geo	Color	Geo
Random objects	23.4	22.4	17.2	18.3
Per-object SDS	32.3	30.5	27.2	25.9
K NeRFs	26.7	25.4	21.0	21.2
+ Per-NeRF losses	27.3	26.1	21.6	22.6
+ Empty NeRF loss	27.7	26.2	22.8	23.2
+ Learn layout	29.9	28.8	24.9	23.5
+ Learn N layouts	**31.3**	**29.9**	**27.1**	**24.8**
Relative layouts	30.4	29.2	25.7	24.0
View dep. prompt	31.0	29.1	25.6	23.6

Figure 4.3: **Evaluating disentanglement and quality.** We optimize a model with $K = 3$ NeRFs on a list of 30 prompts, each containing three objects. We then automatically pair each NeRF with a description of one of the objects in the prompt and report average NeRF-object CLIP score (see text for details). We also generate each of the $30 \times 3 = 90$ objects from the prompt list individually and compute its score with both the corresponding prompt and a random other one, providing upper and lower bounds for performance on this task. Training K NeRFs provides some decomposition, but most objects are scattered across 2 or 3 models. Learning one layout alleviates some of these issues, but only with multiple layouts do we see strong disentanglement. We show two representative examples of emergent objects to visualize these differences.



(a) **Frozen object** (b) **Disentangled objects** (c) **Generated scene**

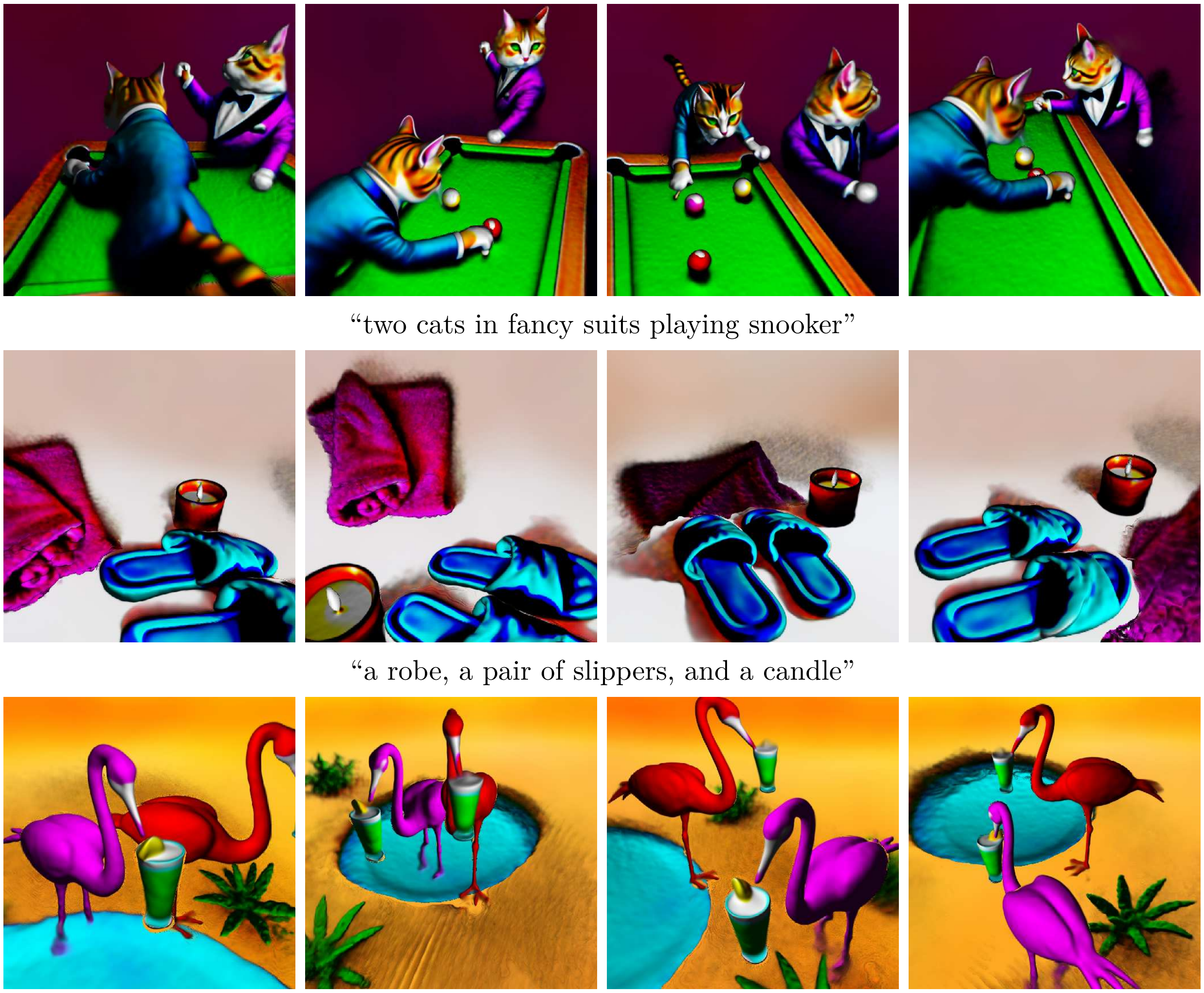
"a cat wearing a hawaiian shirt and sunglasses, having a drink on a beach towel"

"a cat wearing a santa costume holding a present next to a miniature christmas tree"

"a lion in a leather jacket riding a motorcycle between two differently colored cones"

"a modern nightstand with a lamp and a miniature motorcycle model on it, on top of a small rug"

Figure 4.4: **Conditional optimization.** We can take advantage of our structured representation to learn a scene given a 3D asset in addition to a text prompt, such as a specific cat or motorcycle **(a)**. By freezing the NeRF weights but not the layout weights, the model learns to arrange the provided asset in the context of the other objects it discovers **(b)**. We show the entire composite scenes the model creates in **(c)** from two views, along with surface normals and a textureless render.

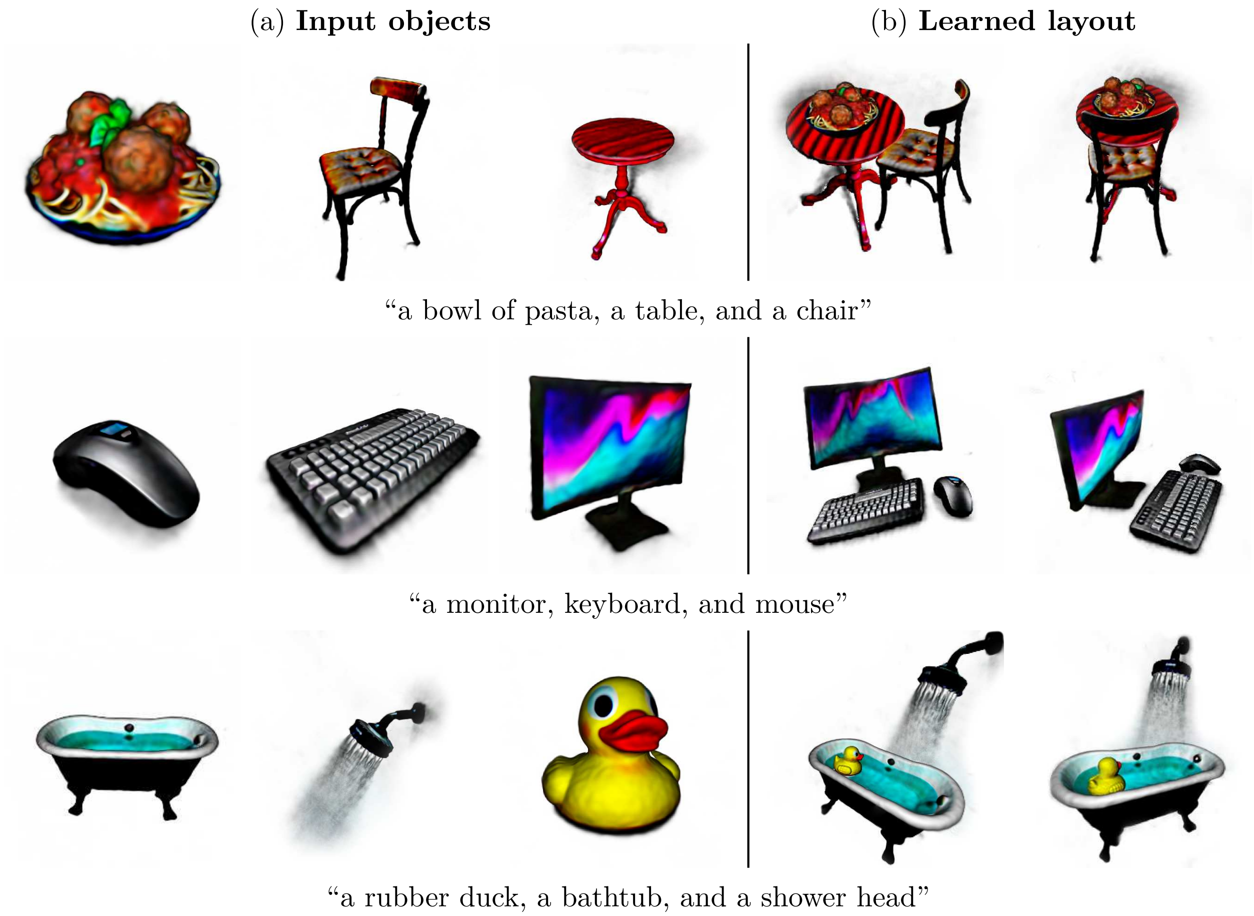


"two cats in fancy suits playing snooker"

"a robe, a pair of slippers, and a candle"

"two flamingos sipping on cocktails in a desert oasis"

Figure 4.5: **Layout diversity.** Our method discovers different plausible arrangements for objects. Here, we optimize each example over $N = 4$ layouts and show differences in composited scenes, *e.g.* flamingos wading inside vs. beside the pond, and cats in different poses around the snooker table.



(a) **Input objects**

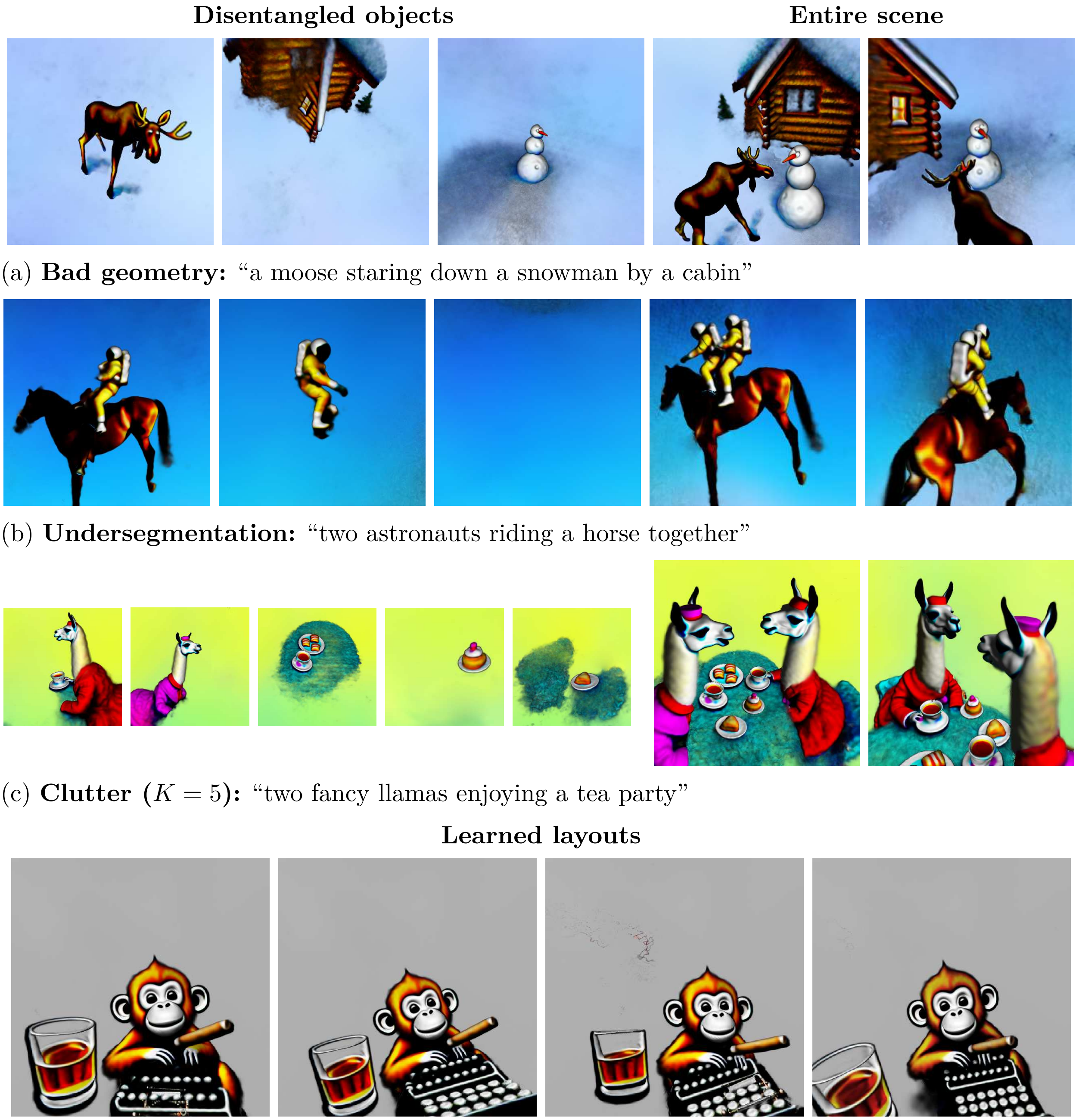
(b) **Learned layout**

“a bowl of pasta, a table, and a chair”

“a monitor, keyboard, and mouse”

“a rubber duck, a bathtub, and a shower head”

Figure 4.6: **Optimizing layout.** Allowing gradients to flow only into layout parameters while freezing a set of provided 3D assets results in reasonable object configurations, such as a chair tucked into a table with spaghetti on it, despite no such guidance being provided in the text conditioning.



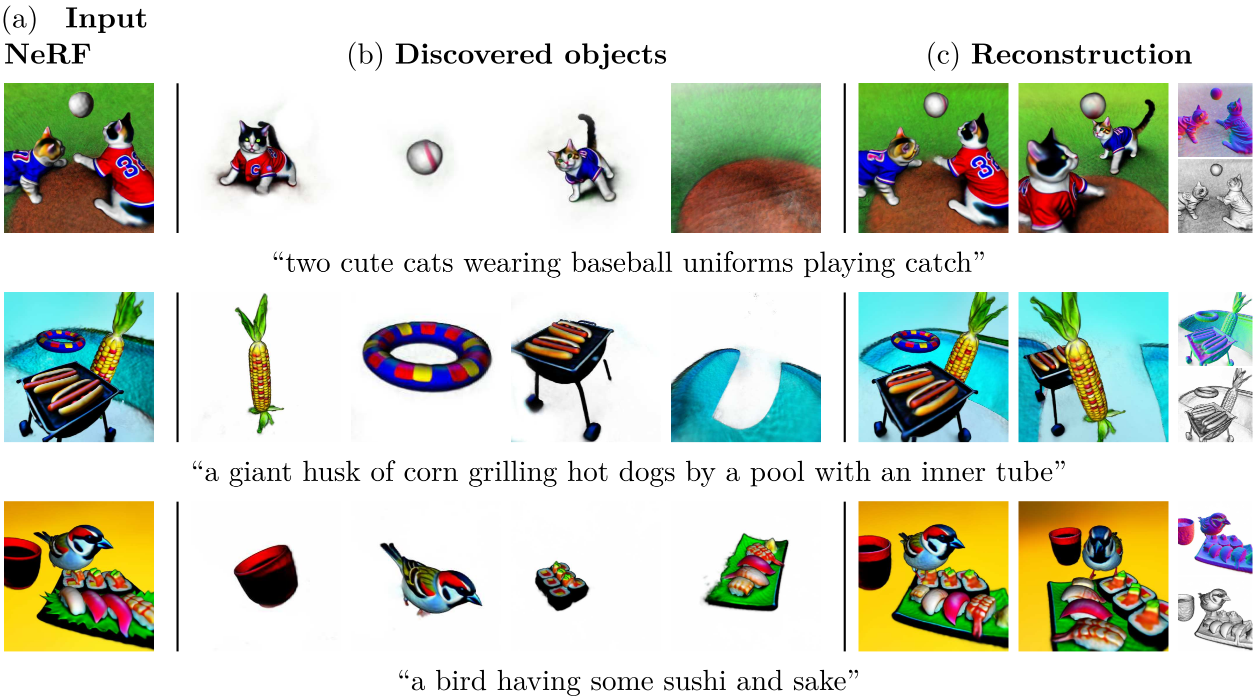
(a) **Bad geometry:** “a moose staring down a snowman by a cabin”

(b) **Undersegmentation:** “two astronauts riding a horse together”

(c) **Clutter** ($K = 5$)**:** “two fancy llamas enjoying a tea party”

(d) **Overly similar layouts:** “a monkey having a whiskey and a cigar, using a typewriter”

Figure 4.7: **Limitations.** Layout learning inherits failure modes from SDS, such as bad geometry of a cabin with oddly intersecting exterior walls **(a)**. It also may undesirably group objects that always move together **(b)** such as a horse and its rider, and **(c)** for certain prompts that generate many small objects, choosing K correctly is challenging, hurting disentanglement. In some cases **(d)**, despite different initial values, layouts converge to very similar final configurations.



(a) **Input NeRF**

(b) **Discovered objects**

(c) **Reconstruction**

“two cute cats wearing baseball uniforms playing catch”

“a giant husk of corn grilling hot dogs by a pool with an inner tube”

“a bird having some sushi and sake”

Figure 4.8: **Decomposing NeRFs of scenes.** Given a NeRF representing a scene **(a)** and a caption, layout learning is able to parse the scene into the objects it contains without any per-object supervision **(b)**. We accomplish this by requiring renders of one of the N learned layouts to match the same view rendered from the target NeRF **(c)**, using a simple L_2 reconstruction loss with $\lambda = 0.05$.

```
# Initialize variables
quat = normal((N, K, 4), mean=[0,0,0,1.], std=0.1)
trans = normal((N, K, 3), mean=0., std=0.3)
scale = normal((N, K, 1), mean=1., std=0.3)
nerfs = [init_nerf() for i in range(K)]

# Transform rays for NeRF k using layout n
def transform(rays, k, n):
  rot = quaternion_to_matrix(quat)
  rays['orig'] = rot[n,k] @ rays['orig'] - trans[n,k]
  rays['orig'] *= scale[n,k]
  rays['dir'] = scale[n,k] * rot[n,k] @ rays['dir']
  return rays

# Composite K NeRFs into one volume
def composite_nerfs(per_nerf_rays):
  per_nerf_out = [nerf(rays) for nerf, rays
    in zip(nerfs, per_nerf_rays)]
  densities = [out['density'] for out in per_nerf_out]
  out = {'density': sum(densities)}
  wts = [d/sum(densities) for d in densities]
  rgbs = [out['rgb'] for out in per_nerf_out]
  out['rgb'] = sum(w*rgb for w,rgb in zip(wts, rgbs))
  return out, per_nerf_out

# Train
optim = shampoo(params=[nerfs, quat, trans, scale])
for step in range(num_steps):
  rays = sample_camera_rays()
  n = random.uniform(N)
  per_nerf_rays = [
    transform(rays, k, n) for k in range(K)
  ]
  vol, per_nerf_vols = composite_nerfs(per_nerf_rays)
  image = render(vol, rays)
  loss = SDS(image, prompt, diffusion_model)
  loss += regularize(per_nerf_vols)
  loss.backward()
  optim.step_and_zero_grad()
```

Figure 4.9: **Pseudocode for layout learning**, with segments inherited from previous work abstracted into functions.

```
def soft_bin(x, t=0.01, eps=1e-7):
  # x has shape (..., H, W)
  bin = sigmoid((x - 0.5)/t)
  min = bin.min(axis=(-1, -2), keepdims=True)
  max = bin.max(axis=(-1, -2), keepdims=True)
  return (bin - min) / (max - min + eps)
soft_bin_acc = soft_bin(acc).mean((-1,-2))
empty_loss = empty_loss_margin - soft_bin_acc
empty_loss = max(empty_loss, 0.)
```

Figure 4.10: **Pseudocode for empty NeRF regularization,** where `soft_bin_acc` computes $\bar{\boldsymbol{\alpha}}_{\text{bin}}$ in Equation 4.5.

```
'a cup of coffee, a croissant, and a closed book',
'a pair of slippers, a robe, and a candle',
'a basket of berries, a carton of whipped cream, and an orange',
'a guitar, a drum set, and an amp',
'a campfire, a bag of marshmallows, and a warm blanket',
'a pencil, an eraser, and a protractor',
'a fork, a knife, and a spoon',
'a baseball, a baseball bat, and a baseball glove',
'a paintbrush, an empty easel, and a palette',
'a teapot, a teacup, and a cucumber sandwich',
'a wallet, keys, and a smartphone',
'a backpack, a water bottle, and a bag of chips',
'a diamond, a ruby, and an emerald',
'a pool table, a dartboard, and a stool',
'a tennis racket, a tennis ball, and a net',
'sunglasses, sunscreen, and a beach towel',
'a ball of yarn, a pillow, and a fluffy cat',
'an old-fashioned typewriter, a cigar, and a glass of whiskey',
'a shovel, a pail, and a sandcastle',
'a microscope, a flask, and a laptop',
'a sunny side up egg, a piece of toast, and some strips of bacon',
'a vase of roses, a slice of chocolate cake, and a bottle of red wine',
'three playing cards, a stack of poker chips, and a flute of champagne',
'a tomato, a stalk of celery, and an onion',
'a coffee machine, a jar of milk, and a pile of coffee beans',
'a bag of flour, a bowl of eggs, and a stick of butter',
'a hot dog, a bottle of soda, and a picnic table',
'a pothos houseplant, an armchair, and a floor lamp',
'an alarm clock, a banana, and a calendar',
'a wrench, a hammer, and a measuring tape',
'a backpack, a bicycle helmet, and a watermelon'
```

Figure 4.11: **Prompts used for CLIP evaluation.** Each prompt is injected into the template "a DSLR photo of {prompt}, plain solid color background". To generate individual objects, the three objects in each prompt are separated into three new prompts and optimized independently.

Chapter 5

Looking Ahead

It is difficult to make predictions, especially about the future.

Niels Bohr, or an old Danish proverb

At some point around halfway through my time in Berkeley, the scale and quality of visual generative models took a huge, seemingly discontinuous leap forward. The dirty secret to this phenomenon is that these models are not really that unsupervised anymore (and perhaps they never were). I refer not only to the importance of data curation in shaping the distribution we ask our model to fit, but also to the reliance on text conditioning, which is abundantly available, compositional, and highly effective at reducing the difficulty of the task at hand. At the asymptote, what does it even mean to draw a sample from $p(x)$ over the entire universe? We bypass this by modeling $p(x|y)$ instead, arguably a significantly more supervised setting.

Throughout the course of my graduate school career, I have thus come to shed any attachment to distinctions about "true" [37] unsupervised learning, especially as it relates to language. Of course, I do believe that scalable approaches will continue to rely minimally on manual annotation, but the opposition to annotations is pragmatic rather than ideological. On a higher level, intelligent agents should eventually operate on all modalities simultaneously, and language is a crucial interface for us humans to interact with the models we train.

Still, language is undoubtedly an incomplete and inadequate interface on its own. Turning briefly to biology, many crucial forms of communication among humans and other species are non-verbal [39, 27, 86]. As for machine learning, the astute reader will have noticed that all the forms of control (and thus, demonstrations of knowledge) presented in this thesis are much more effectively conveyed through a *non-textual interface.*

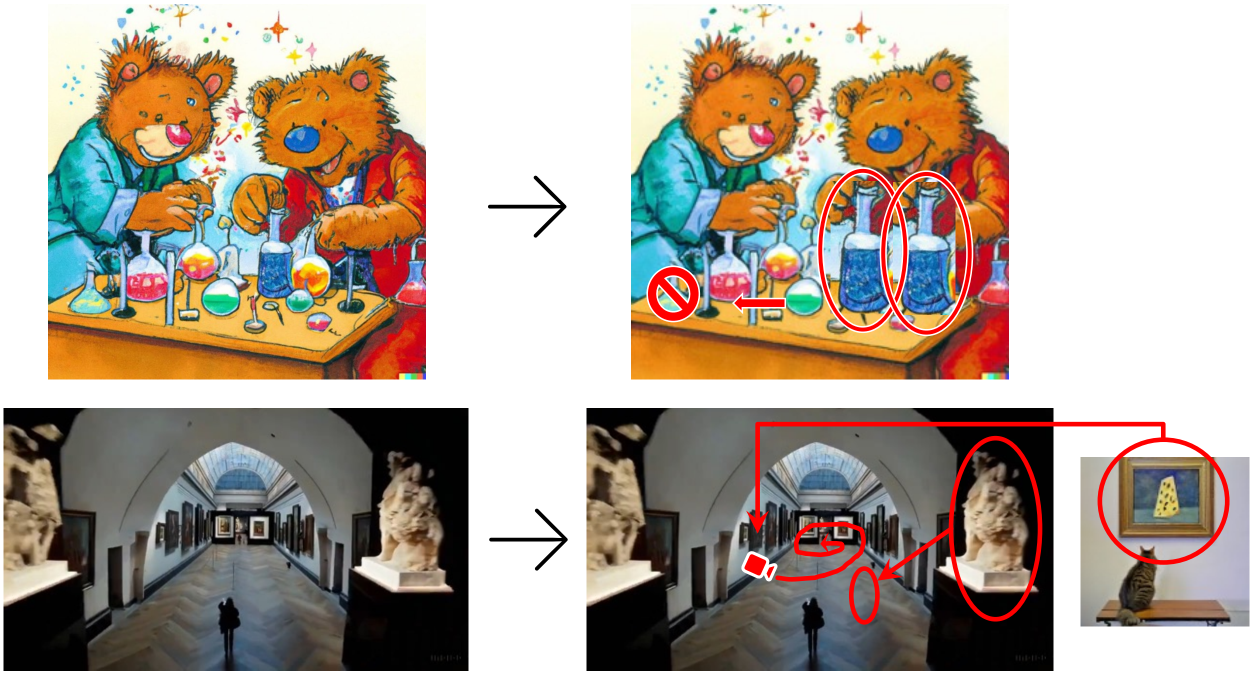


Figure 5.1: ***An* output versus *the* output.** Top: An image generated by DALL-E 2 [135] given the prompt `teddy bears mixing sparkling chemicals as mad scientists in the style of a 1990s Saturday morning cartoon`. Bottom: A frame from a video generated by Sora [173], given the prompt `fly through tour of a museum with many paintings and sculptures and beautiful works of art in all styles`. The models we are sampling from (left) are powerful, but because their interface is so limited, we are unable to do things like move, delete, resize, or restyle specific elements of their outputs (right).

5.1 Beyond text

As the complexity of visual data we are able to model increases, more and more becomes challenging to express through text (Figure 5.1). If image and video generators are indeed to serve as drop-in world models, usable in creative contexts as well as (more tantalizingly) for agents to plan their actions from observations of their surroundings, they must be able to perform the sort of counterfactual manipulation and disentanglement explored in this document. It does not suffice to create an aesthetically pleasing output aligned with a text prompt. Though techniques such as recaptioning [13] expand the scope of what we can address through text alone, they are still doomed to fall short due to the limitations of language itself. Perhaps we require a novel form of conditioning that is non-linguistic, yet still a meta-interface over different forms of desired control.

The approaches discussed in this thesis are post-hoc, discovering new ways to use an off-the-shelf text-to-image model. Though this constraint was partially imposed by logistical issues preventing a random graduate student from training multi-billion parameter models at a mega-corporation, this order of operations seems natural in deep learning given the black-box nature of our toolkit. First design a model, then gather data for it, then train it, then uncover and leverage the structure that it learns. One cannot help but wonder: Will the loop ever close? Will our valiant attempts to understand and extract the knowledge these models acquire inform subsequent generations of model design?

I leave these questions to future work.

Bibliography

[1] Rameen Abdal, Yipeng Qin, and Peter Wonka. “Image2stylegan: How to embed images into the stylegan latent space?” In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 4432–4441.

[2] Rameen Abdal et al. “Styleflow: Attribute-conditioned exploration of stylegan-generated images using conditional continuous normalizing flows”. In: *ACM Transactions on Graphics (TOG)* 40.3 (2021), pp. 1–21.

[3] Badour AlBahar et al. “Pose with Style: Detail-Preserving Pose-Guided Image Synthesis with Conditional StyleGAN”. In: *ACM Transactions on Graphics* (2021).

[4] Arpit Bansal et al. “Universal Guidance for Diffusion Models”. In: *arXiv preprint arXiv:2302.07121* (2023).

[5] Omer Bar-Tal et al. “Text2live: Text-driven layered image and video editing”. In: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XV*. Springer. 2022, pp. 707–723.

[6] Connelly Barnes et al. “PatchMatch: a randomized correspondence algorithm for structural image editing”. In: *ACM Trans. Graph.* 28.3 (2009), p. 24.

[7] Jonathan T. Barron et al. “Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 5470–5479.

[8] Jonathan T. Barron et al. “Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 5855–5864.

[9] David Bau et al. “Gan dissection: Visualizing and understanding generative adversarial networks”. In: *arXiv preprint arXiv:1811.10597* (2018).

[10] David Bau et al. “Rewriting a deep generative model”. In: *European Conference on Computer Vision*. Springer. 2020, pp. 351–369.

[11] David Bau et al. “Seeing what a gan cannot generate”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 4502–4511.

[12] Daniel Bear et al. “Learning physical graph representations from visual scenes”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 6027–6039.

[13] James Betker et al. "Improving image generation with better captions". In: *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf* 2.3 (2023), p. 8.

[14] Irving Biederman. *On the semantics of a glance at a scene.* Routledge, 1981, pp. 213–253.

[15] Irving Biederman, Robert J Mezzanotte, and Jan C Rabinowitz. "Scene perception: Detecting and judging objects undergoing relational violations". In: *Cognitive psychology* 14.2 (1982), pp. 143–177.

[16] Andrew Brock, Jeff Donahue, and Karen Simonyan. "Large Scale GAN Training for High Fidelity Natural Image Synthesis". In: *ArXiv* abs/1809.11096 (2018).

[17] Andrew Brock, Jeff Donahue, and Karen Simonyan. "Large scale gan training for high fidelity natural image synthesis". In: 2018.

[18] Tim Brooks and Alexei A Efros. "Hallucinating Pose-Compatible Scenes". In: *arXiv preprint arXiv:2112.06909* (2021).

[19] Tim Brooks, Aleksander Holynski, and Alexei A Efros. "Instructpix2pix: Learning to follow image editing instructions". In: *arXiv preprint arXiv:2211.09800* (2022).

[20] Chad Carson et al. "Blobworld: A system for region-based image indexing and retrieval". In: *International conference on advances in visual information systems.* Springer. 1999, pp. 509–517.

[21] Lucy Chai, Jonas Wulff, and Phillip Isola. "Using latent space regression to analyze and leverage compositionality in gans". In: *arXiv preprint arXiv:2103.10426* (2021).

[22] Huiwen Chang et al. "Muse: Text-to-image generation via masked generative transformers". In: *ICML.* 2023.

[23] Minghao Chen, Iro Laina, and Andrea Vedaldi. "Training-Free Layout Control with Cross-Attention Guidance". In: *arXiv preprint arXiv:2304.03373* (2023).

[24] Qifeng Chen and Vladlen Koltun. "Photographic image synthesis with cascaded refinement networks". In: *Proceedings of the IEEE international conference on computer vision.* 2017, pp. 1511–1520.

[25] Dana Cohen-Bar et al. "Set-the-Scene: Global-Local Training for Generating Controllable NeRF Scenes". In: *ICCV.* 2023.

[26] Edo Collins et al. "Editing in style: Uncovering the local semantics of gans". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2020, pp. 5771–5780.

[27] Daniel C Dennett. *The intentional stance.* MIT press, 1989.

[28] Emily L Denton, Soumith Chintala, Rob Fergus, et al. "Deep generative image models using a laplacian pyramid of adversarial networks". In: *Advances in neural information processing systems* 28 (2015).

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

[30] Prafulla Dhariwal and Alex Nichol. “Diffusion Models Beat GANs on Image Synthesis”. In: *ArXiv* abs/2105.05233 (2021).

[31] Dave Epstein, Boyuan Chen, and Carl Vondrick. “Oops! predicting unintentional action in video”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 919–929.

[32] Dave Epstein and Carl Vondrick. “Learning goals from failure”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 11194–11204.

[33] Dave Epstein et al. “Blobgan: Spatially disentangled scene representations”. In: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XV*. Springer. 2022, pp. 616–635.

[34] Dave Epstein et al. “Diffusion Self-Guidance for Controllable Image Generation”. In: *Advances in Neural Information Processing Systems*. 2023.

[35] Dave Epstein et al. *Disentangled 3D Scene Generation with Layout Learning*. 2024. arXiv: `2402.16936 [cs.CV]`.

[36] Dave Epstein et al. “Learning temporal dynamics from cycles in narrated video”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 1480–1489.

[37] A. Flew. *God & Philosophy*. Harcourt, Brace & World, 1966. URL: `https://books.google.com/books?id=K7YXAAAAIAAJ`.

[38] Rinon Gal et al. “An image is worth one word: Personalizing text-to-image generation using textual inversion”. In: *arXiv preprint arXiv:2208.01618* (2022).

[39] Shiry Sara Ginosar. *Modeling Visual Minutiae: Gestures, Styles, and Temporal Patterns*. University of California, Berkeley, 2020.

[40] Lore Goetschalckx et al. “Ganalyze: Toward visual definitions of cognitive image properties”. In: *Proceedings of the ieee/cvf international conference on computer vision*. 2019, pp. 5744–5753.

[41] Ian Goodfellow et al. “Generative adversarial nets”. In: *Advances in neural information processing systems* 27 (2014).

[42] Alexandros Graikos et al. “Diffusion models as plug-and-play priors”. In: *arXiv:2206.09012* (2022).

[43] Abhinav Gupta, Alexei A Efros, and Martial Hebert. “Blocks world revisited: Image understanding using qualitative geometry and mechanics”. In: *European Conference on Computer Vision*. Springer. 2010, pp. 482–496.

[44] Vineet Gupta, Tomer Koren, and Yoram Singer. "Shampoo: Preconditioned Stochastic Tensor Optimization". In: *ICML*. 2018. eprint: `1802.09568`.

[45] Bharath Hariharan et al. "Simultaneous detection and segmentation". In: *European conference on computer vision*. Springer. 2014, pp. 297–312.

[46] Erik Härkönen et al. "Ganspace: Discovering interpretable gan controls". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 9841–9850.

[47] Kaiming He et al. "Masked autoencoders are scalable vision learners". In: *arXiv preprint arXiv:2111.06377* (2021).

[48] Xingzhe He, Bastian Wandt, and Helge Rhodin. "Latentkeypointgan: Controlling gans via latent keypoints". In: *arXiv preprint arXiv:2103.15812* (2021).

[49] Varsha Hedau, Derek Hoiem, and David Forsyth. "Recovering the spatial layout of cluttered rooms". In: *2009 IEEE 12th international conference on computer vision*. IEEE, pp. 1849–1856.

[50] Eric Hedlin et al. "Unsupervised Semantic Correspondence Using Stable Diffusion". In: *arXiv preprint arXiv:2305.15581* (2023).

[51] Olivier J Hénaff et al. "Object discovery and representation networks". In: *European Conference on Computer Vision*. Springer. 2022, pp. 123–143.

[52] Amir Hertz et al. "Prompt-to-prompt image editing with cross attention control". In: *arXiv preprint arXiv:2208.01626* (2022).

[53] Martin Heusel et al. "Gans trained by a two time-scale update rule converge to a local nash equilibrium". In: *Advances in neural information processing systems* 30 (2017).

[54] Irina Higgins et al. "beta-vae: Learning basic visual concepts with a constrained variational framework". In: (2016).

[55] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising Diffusion Probabilistic Models". In: *NeurIPS* (2020). Ed. by H. Larochelle et al.

[56] Jonathan Ho and Tim Salimans. "Classifier-free diffusion guidance". In: *arXiv:2207.12598* (2022).

[57] Howard S Hock et al. "Real-world schemata and scene recognition in adults and children". In: *Memory & Cognition* 6.4 (1978), pp. 423–431.

[58] Almut Hoffmann, Vanessa Rüttler, and Andreas Nieder. "Ontogeny of object permanence and object tracking in the carrion crow, Corvus corone". In: *Animal behaviour* 82.2 (2011), pp. 359–367.

[59] Jordan Hoffmann et al. "Training compute-optimal large language models". In: *arXiv preprint arXiv:2203.15556* (2022).

[60] Derek Hoiem, Alexei A. Efros, and Martial Hebert. "Recovering Surface Layout from an Image". In: *IJCV* 75.1 (Oct. 2007), pp. 151–172.

[61] Xun Huang et al. “Multimodal unsupervised image-to-image translation”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 172–189.

[62] Phillip Isola and Ce Liu. “Scene Collaging: Analysis and Synthesis of Natural Images with Semantic Layers”. In: *ICCV*. 2013.

[63] Phillip Isola et al. “Image-to-image translation with conditional adversarial networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1125–1134.

[64] Allan Jabri, David Fleet, and Ting Chen. “Scalable Adaptive Computation for Iterative Generation”. In: *ICML*. 2023.

[65] Andrew Jaegle et al. “Perceiver io: A general architecture for structured inputs & outputs”. In: *arXiv preprint arXiv:2107.14795* (2021).

[66] Andrew Jaegle et al. “Perceiver: General perception with iterative attention”. In: *International conference on machine learning*. PMLR. 2021, pp. 4651–4664.

[67] Ali Jahanian, Lucy Chai, and Phillip Isola. “On the” steerability” of generative adversarial networks”. In: *arXiv preprint arXiv:1907.07171* (2019).

[68] Ajay Jain et al. “Zero-Shot Text-Guided Object Generation with Dream Fields”. In: *CVPR*. 2022.

[69] 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.

[70] Justin Johnson et al. “Clevr: A diagnostic dataset for compositional language and elementary visual reasoning”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2901–2910.

[71] Jared Kaplan et al. “Scaling laws for neural language models”. In: *arXiv preprint arXiv:2001.08361* (2020).

[72] Tero Karras, Samuli Laine, and Timo Aila. “A style-based generator architecture for generative adversarial networks”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 4401–4410.

[73] Tero Karras et al. “Alias-free generative adversarial networks”. In: *Advances in Neural Information Processing Systems* 34 (2021).

[74] Tero Karras et al. “Analyzing and improving the image quality of stylegan”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 8110–8119.

[75] Tero Karras et al. “Analyzing and improving the image quality of stylegan”. In: 2020.

[76] Tero Karras et al. “Progressive growing of gans for improved quality, stability, and variation”. In: 2018.

[77] Bahjat Kawar et al. “Imagic: Text-based real image editing with diffusion models”. In: *arXiv preprint arXiv:2210.09276* (2022).

[78] Bingxin Ke et al. “Repurposing Diffusion-Based Image Generators for Monocular Depth Estimation”. In: *arXiv preprint arXiv:2312.02145* (2023).

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

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

[81] Diederik P Kingma et al. “Variational Diffusion Models”. In: *NeurIPS* (2021).

[82] Alexander Kirillov et al. “Segment anything”. In: *arXiv preprint arXiv:2304.02643* (2023).

[83] Mingi Kwon, Jaeseok Jeong, and Youngjung Uh. “Diffusion models already have a semantic latent space”. In: *arXiv preprint arXiv:2210.10960* (2022).

[84] Tuomas Kynkäänniemi et al. “Improved precision and recall metric for assessing generative models”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[85] MF Land and D Nilsson. “The origin of vision”. In: *Animal Eyes* 591 (2012), pp. 1–15.

[86] Clarence Irving Lewis. *Mind and the world-order: Outline of a theory of knowledge*. Courier Corporation, 1956.

[87] Kathleen M Lewis, Srivatsan Varadharajan, and Ira Kemelmacher-Shlizerman. “TryOnGAN: Body-Aware Try-On via Layered Interpolation”. In: *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2021)* 40.4 (2021).

[88] Ang Li et al. “Learning Visual N-Grams from Web Data”. In: *ICCV*. 2017.

[89] Ke Li, Tianhao Zhang, and Jitendra Malik. “Diverse image synthesis from semantic layouts via conditional imle”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 4220–4229.

[90] Yanghao Li et al. “Exploring plain vision transformer backbones for object detection”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 280–296.

[91] Yuheng Li et al. “Collaging Class-specific GANs for Semantic Image Synthesis”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 14418–14427.

[92] Chen-Hsuan Lin et al. “BARF: Bundle-Adjusting Neural Radiance Fields”. In: *ICCV*. 2021.

[93] Chen-Hsuan Lin et al. “Magic3d: High-resolution text-to-3d content creation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 300–309.

[94] Nan Liu et al. “Compositional Visual Generation with Composable Diffusion Models”. In: *European Conference on Computer Vision*. 2022.

[95] Ruoshi Liu et al. “Zero-1-to-3: Zero-shot one image to 3d object”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 9298–9309.

[96] Vivian Liu and Lydia B Chilton. “Design guidelines for prompt engineering text-to-image generative models”. In: *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. 2022, pp. 1–23.

[97] Francesco Locatello et al. “Challenging common assumptions in the unsupervised learning of disentangled representations”. In: *international conference on machine learning*. PMLR. 2019, pp. 4114–4124.

[98] Francesco Locatello et al. “Object-centric learning with slot attention”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 11525–11538.

[99] Grace Luo et al. “Diffusion Hyperfeatures: Searching Through Time and Space for Semantic Correspondence”. In: *arXiv preprint arXiv:2305.14334* (2023).

[100] Tomasz Malisiewicz and Alyosha Efros. “Beyond categories: The visual memex model for reasoning about object relationships”. In: *Advances in neural information processing systems* 22 (2009).

[101] Charles R Marshall. “Explaining the Cambrian “explosion” of animals”. In: *Annu. Rev. Earth Planet. Sci.* 34 (2006), pp. 355–384.

[102] Youssef A Mejjati et al. “GaussiGAN: Controllable Image Synthesis with 3D Gaussians from Unposed Silhouettes”. In: *arXiv preprint arXiv:2106.13215* (2021).

[103] Youssef A. Mejjati et al. “Generating Object Stamps”. In: *Computer Vision and Pattern Recognition Workshop on AI for Content Creation (CVPRW)*. June 2020.

[104] Chenlin Meng et al. “Sdedit: Guided image synthesis and editing with stochastic differential equations”. In: *International Conference on Learning Representations*. 2021.

[105] Chenlin Meng et al. “Sdedit: Image synthesis and editing with stochastic differential equations”. In: *arXiv preprint arXiv:2108.01073* (2021).

[106] Ben Mildenhall et al. “NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis”. In: *ECCV* (2020).

[107] Ron Mokady et al. “Null-text Inversion for Editing Real Images using Guided Diffusion Models”. In: *arXiv preprint arXiv:2211.09794* (2022).

[108] Tom Monnier et al. “Differentiable Blocks World: Qualitative 3D Decomposition by Rendering Primitives”. In: *Neural Information Processing Systems*. 2023.

[109] Thomas Müller et al. “Instant Neural Graphics Primitives with a Multiresolution Hash Encoding”. In: *ACM Trans. Graph.* 41.4 (July 2022), 102:1–102:15. DOI: 10.1145/3528223.3530127. URL: https://doi.org/10.1145/3528223.3530127.

[110] Thu Nguyen-Phuoc et al. “Hologan: Unsupervised learning of 3d representations from natural images”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 7588–7597.

[111] Thu H Nguyen-Phuoc et al. "Blockgan: Learning 3d object-aware scene representations from unlabelled images". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 6767–6778.

[112] Alex Nichol et al. "GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models". In: *ICML*. 2022.

[113] Alex Nichol et al. "Glide: Towards photorealistic image generation and editing with text-guided diffusion models". In: *arXiv preprint arXiv:2112.10741* (2021).

[114] Alex Nichol et al. "Glide: Towards photorealistic image generation and editing with text-guided diffusion models". In: *arXiv preprint arXiv:2112.10741* (2021).

[115] Michael Niemeyer and Andreas Geiger. "GIRAFFE: Representing Scenes as Compositional Generative Neural Feature Fields". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[116] Michael Niemeyer and Andreas Geiger. "Giraffe: Representing scenes as compositional generative neural feature fields". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 11453–11464.

[117] Mark Nitzberg and David Bryant Mumford. *The 2.1-D sketch*. IEEE Computer Society Press, 1990.

[118] Yu-ichi Ohta, Takeo Kanade, and Toshiyuki Sakai. "An analysis system for scenes containing objects with substructures". In: *Proceedings of the Fourth International Joint Conference on Pattern Recognitions*. 1978, pp. 752–754.

[119] Y. Ohta, Takeo Kanade, and T. Sakai. "An Analysis System for Scenes Containing objects with Substructures". In: *Proceedings of 4th International Joint Conference on Pattern Recognition (IJCPR '78)*. Nov. 1978, pp. 752–754.

[120] Deniz Oktay, Carl Vondrick, and Antonio Torralba. *Counterfactual Image Networks*. 2018. URL: `https://openreview.net/forum?id=SyYYPdg0-`.

[121] Aude Oliva and Antonio Torralba. "The role of context in object recognition". In: *Trends in cognitive sciences* 11.12 (2007), pp. 520–527.

[122] Dong Huk Park et al. "Benchmark for compositional text-to-image synthesis". In: *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*. 2021.

[123] Taesung Park et al. "Semantic image synthesis with spatially-adaptive normalization". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 2337–2346.

[124] Taesung Park et al. "Swapping autoencoder for deep image manipulation". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 7198–7211.

[125] Or Patashnik et al. "Styleclip: Text-driven manipulation of stylegan imagery". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 2085–2094.

[126] Deepak Pathak et al. "Context Encoders: Feature Learning by Inpainting". In: 2016.

[127] William Peebles et al. "The hessian penalty: A weak prior for unsupervised disentanglement". In: *European Conference on Computer Vision*. Springer. 2020, pp. 581–597.

[128] Jean Piaget, Margaret Cook, et al. *The origins of intelligence in children*. Vol. 8. 5. International Universities Press New York, 1952.

[129] Ryan Po and Gordon Wetzstein. "Compositional 3d scene generation using locally conditioned diffusion". In: *arXiv preprint arXiv:2303.12218* (2023).

[130] Ben Poole et al. "Dreamfusion: Text-to-3d using 2d diffusion". In: *arXiv preprint arXiv:2209.14988* (2022).

[131] Thomas Porter and Tom Duff. "Compositing digital images". In: *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*. 1984, pp. 253–259.

[132] Konpat Preechakul et al. "Diffusion autoencoders: Toward a meaningful and decodable representation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 10619–10629.

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

[134] Alec Radford et al. "Learning transferable visual models from natural language supervision". In: *International conference on machine learning*. PMLR. 2021, pp. 8748–8763.

[135] Aditya Ramesh et al. "Hierarchical Text-Conditional Image Generation with CLIP Latents". In: *arXiv preprint arXiv:2204.06125* (2022). DOI: `10.48550/ARXIV.2204.06125`. URL: `https://arxiv.org/abs/2204.06125`.

[136] Aditya Ramesh et al. "Hierarchical text-conditional image generation with clip latents". In: *arXiv preprint arXiv:2204.06125* 1.2 (2022), p. 3.

[137] Aditya Ramesh et al. "Zero-shot text-to-image generation". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 8821–8831.

[138] Scott Reed et al. "Generative adversarial text to image synthesis". In: *International conference on machine learning*. PMLR. 2016, pp. 1060–1069.

[139] Elad Richardson et al. "Encoding in style: a stylegan encoder for image-to-image translation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 2287–2296.

[140] Lawrence G Roberts. "Machine perception of three-dimensional solids". PhD thesis. Massachusetts Institute of Technology, 1963.

[141] Daniel Roich et al. "Pivotal tuning for latent-based editing of real images". In: *arXiv preprint arXiv:2106.05744* (2021).

[142] Robin Rombach et al. "High-Resolution Image Synthesis with Latent Diffusion Models". In: *arXiv preprint arXiv:2112.10752* (2021).

[143] Robin Rombach et al. "High-resolution image synthesis with latent diffusion models". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 10684–10695.

[144] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *ArXiv* abs/1505.04597 (2015).

[145] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. "SinGAN: Learning a Generative Model from a Single Natural Image". In: *Computer Vision (ICCV), IEEE International Conference on*. 2019.

[146] Michael Rubinstein et al. "Unsupervised joint object discovery and segmentation in internet images". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2013, pp. 1939–1946.

[147] Nataniel Ruiz et al. "Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation". In: *arXiv preprint arXiv:2208.12242* (2022).

[148] Bryan Russell et al. "Segmenting scenes by matching image composites". In: *Advances in Neural Information Processing Systems* 22 (2009).

[149] Bryan C Russell et al. "Using multiple segmentations to discover objects and their extent in image collections". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 2. IEEE. 2006, pp. 1605–1614.

[150] Chitwan Saharia et al. "Palette: Image-to-image diffusion models". In: *arXiv preprint arXiv:2111.05826* (2021).

[151] Chitwan Saharia et al. "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding". In: *arXiv:2205.11487* (2022).

[152] Chitwan Saharia et al. "Photorealistic text-to-image diffusion models with deep language understanding". In: *arXiv preprint arXiv:2205.11487* 35 (2022), pp. 36479–36494.

[153] Mehdi SM Sajjadi et al. "Object scene representation transformer". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 9512–9524.

[154] Kripasindhu Sarkar et al. *Style and Pose Control for Image Synthesis of Humans from a Single Monocular View*. 2021. arXiv: `2102.11263 [cs.CV]`.

[155] Evan Shelhamer, Dequan Wang, and Trevor Darrell. "Blurring the line between structure and learning to optimize and adapt receptive fields". In: *arXiv preprint arXiv:1904.11487* (2019).

[156] Yujun Shen and Bolei Zhou. “Closed-form factorization of latent semantics in gans”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2021, pp. 1532–1540.

[157] Yujun Shen et al. “Interfacegan: Interpreting the disentangled face representation learned by gans”. In: *IEEE transactions on pattern analysis and machine intelligence* (2020).

[158] Jianbo Shi and Jitendra Malik. “Normalized cuts and image segmentation”. In: *IEEE Transactions on pattern analysis and machine intelligence* 22.8 (2000), pp. 888–905.

[159] Aliaksandr Siarohin et al. “First Order Motion Model for Image Animation”. In: *Conference on Neural Information Processing Systems (NeurIPS).* Dec. 2019.

[160] Aliaksandr Siarohin et al. “Motion Representations for Articulated Animation”. In: *CVPR.* 2021.

[161] Nathan Silberman et al. “Indoor segmentation and support inference from rgbd images”. In: *European conference on computer vision.* Springer. 2012, pp. 746–760.

[162] Denis Simakov et al. “Summarizing visual data using bidirectional similarity”. In: *CVPR.* IEEE Computer Society, 2008.

[163] Cameron Smith et al. “Unsupervised discovery and composition of object light fields”. In: *arXiv preprint arXiv:2205.03923* (2022).

[164] Jascha Sohl-Dickstein et al. “Deep Unsupervised Learning using Nonequilibrium Thermodynamics”. In: *ICML* (2015).

[165] Jiaming Song, Chenlin Meng, and Stefano Ermon. “Denoising Diffusion Implicit Models”. In: *CoRR* abs/2010.02502 (2020). arXiv: `2010.02502`. URL: `https://arxiv.org/abs/2010.02502`.

[166] Yang Song et al. “Score-Based Generative Modeling through Stochastic Differential Equations”. In: *ICLR* (2021).

[167] Elizabeth S Spelke. “Principles of object perception”. In: *Cognitive science* 14.1 (1990), pp. 29–56.

[168] Robin Strudel et al. “Segmenter: Transformer for semantic segmentation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 2021, pp. 7262–7272.

[169] Erik B Sudderth et al. “Learning hierarchical models of scenes, objects, and parts”. In: *Tenth IEEE International Conference on Computer Vision (ICCV’05) Volume 1.* Vol. 2. IEEE. 2005, pp. 1331–1338.

[170] Dídac Surís, Dave Epstein, and Carl Vondrick. “Globetrotter: Connecting languages by connecting images”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2022, pp. 16474–16484.

[171] Richard Sutton. “The bitter lesson”. In: *Incomplete Ideas (blog)* 13.1 (2019), p. 38.

[172] Matthew Tancik et al. “Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains”. In: *Neural Information Processing Systems*. 2020.

[173] OpenAI Team. 2024. URL: `https://openai.com/research/video-generation-models-as-world-simulators`.

[174] Antonio Torralba et al. “Describing visual scenes using transformed dirichlet processes”. In: *Advances in neural information processing systems* 18 (2005).

[175] Omer Tov et al. “Designing an encoder for stylegan image manipulation”. In: *ACM Transactions on Graphics (TOG)* 40.4 (2021), pp. 1–14.

[176] Zhuowen Tu et al. “Image parsing: Unifying segmentation, detection, and recognition”. In: *International Journal of computer vision* 63.2 (2005), pp. 113–140.

[177] Narek Tumanyan et al. “Plug-and-Play Diffusion Features for Text-Driven Image-to-Image Translation”. In: *arXiv preprint arXiv:2211.12572* (2022).

[178] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. “YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 7464–7475.

[179] Haochen Wang et al. “Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 12619–12629.

[180] Jianyuan Wang et al. “Improving GAN Equilibrium by Raising Spatial Awareness”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 11285–11293.

[181] 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, pp. 8798–8807.

[182] Zhengyi Wang et al. “ProlificDreamer: High-Fidelity and Diverse Text-to-3D Generation with Variational Score Distillation”. In: *arXiv preprint arXiv:2305.16213* (2023).

[183] Max Wertheimer. “Laws of organization in perceptual forms.” In: (1938).

[184] Teresa Wilcox. “Object individuation: Infants’ use of shape, size, pattern, and color”. In: *Cognition* 72.2 (1999), pp. 125–166.

[185] Rundi Wu et al. “ReconFusion: 3D Reconstruction with Diffusion Priors”. In: *arXiv preprint arXiv:2312.02981* (2023).

[186] Zongze Wu, Dani Lischinski, and Eli Shechtman. “Stylespace analysis: Disentangled controls for stylegan image generation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 12863–12872.

[187] Jonas Wulff and Antonio Torralba. "Improving inversion and generation diversity in stylegan using a gaussianized latent space". In: *arXiv preprint arXiv:2009.06529* (2020).

[188] Weihao Xia et al. "GAN inversion: A survey". In: *arXiv preprint arXiv:2101.05278* (2021).

[189] Yoram Yakimovsky and Jerome A. Feldman. "A Semantics-Based Decision Theory Region Analyser". In: *IJCAI*. William Kaufmann, 1973, pp. 580–588.

[190] Ceyuan Yang, Yujun Shen, and Bolei Zhou. "Semantic hierarchy emerges in deep generative representations for scene synthesis". In: *International Journal of Computer Vision* 129.5 (2021), pp. 1451–1466.

[191] Yue Yang et al. "Holodeck: Language Guided Generation of 3D Embodied AI Environments". In: *arXiv preprint arXiv:2312.09067* (2023).

[192] Shunyu Yao et al. "3d-aware scene manipulation via inverse graphics". In: *Advances in neural information processing systems* 31 (2018).

[193] Vickie Ye et al. "Deformable sprites for unsupervised video decomposition". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 2657–2666.

[194] Fisher Yu et al. "Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop". In: *arXiv preprint arXiv:1506.03365* (2015).

[195] Hong-Xing Yu, Leonidas J Guibas, and Jiajun Wu. "Unsupervised discovery of object radiance fields". In: *arXiv preprint arXiv:2107.07905* (2021).

[196] Jiahui Yu et al. "Scaling Autoregressive Models for Content-Rich Text-to-Image Generation". In: *arXiv:2206.10789* (2022).

[197] Jiahui Yu et al. "Scaling autoregressive models for content-rich text-to-image generation". In: *arXiv preprint arXiv:2206.10789* 2.3 (2022), p. 5.

[198] Stella X Yu, Ralph Gross, and Jianbo Shi. "Concurrent object recognition and segmentation by graph partitioning". In: *Advances in neural information processing systems* 15 (2002).

[199] Chen Zhang, Yinghao Xu, and Yujun Shen. "Decorating your own bedroom: Locally controlling image generation with generative adversarial networks". In: *arXiv preprint arXiv:2105.08222* (2021).

[200] Han Zhang et al. "Self-attention generative adversarial networks". In: *International conference on machine learning*. PMLR. 2019, pp. 7354–7363.

[201] 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.

[202] Lvmin Zhang and Maneesh Agrawala. “Adding conditional control to text-to-image diffusion models”. In: *arXiv preprint arXiv:2302.05543* (2023).

[203] Qihang Zhang et al. “SceneWiz3D: Towards Text-guided 3D Scene Composition”. In: *arXiv preprint arXiv:2312.08885* (2023).

[204] Richard Zhang et al. “The unreasonable effectiveness of deep features as a perceptual metric”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 586–595.

[205] Jiapeng Zhu et al. “In-domain gan inversion for real image editing”. In: *European conference on computer vision*. Springer. 2020, pp. 592–608.

[206] Jiapeng Zhu et al. “Region-Based Semantic Factorization in GANs”. In: *arXiv preprint arXiv:2202.09649* (2022).

[207] Jun-Yan Zhu et al. “Generative visual manipulation on the natural image manifold”. In: *European conference on computer vision*. Springer. 2016, pp. 597–613.

[208] Jun-Yan Zhu et al. “Toward multimodal image-to-image translation”. In: *Advances in neural information processing systems* 30 (2017).