LayoutVAE: Stochastic Scene Layout Generation From a Label Set

Akash Abdu Jyothi[1,3], Thibaut Durand[1,3], Jiawei He[1,3], Leonid Sigal[2,3], Greg Mori[1,3]
[1]Simon Fraser University [2]University of British Columbia [3]Borealis AI
{aabdujyo, tdurand, jha203}@sfu.ca lsigal@cs.ubc.ca mori@cs.sfu.ca

Abstract

Recently there is an increasing interest in scene generation within the research community. However, models used for generating scene layouts from textual description largely ignore plausible visual variations within the structure dictated by the text. We propose LayoutVAE, a variational autoencoder based framework for generating stochastic scene layouts. LayoutVAE is a versatile modeling framework that allows for generating full image layouts given a label set, or per label layouts for an existing image given a new label. In addition, it is also capable of detecting unusual layouts, potentially providing a way to evaluate layout generation problem. Extensive experiments on MNIST-Layouts and challenging COCO 2017 Panoptic dataset verifies the effectiveness of our proposed framework.

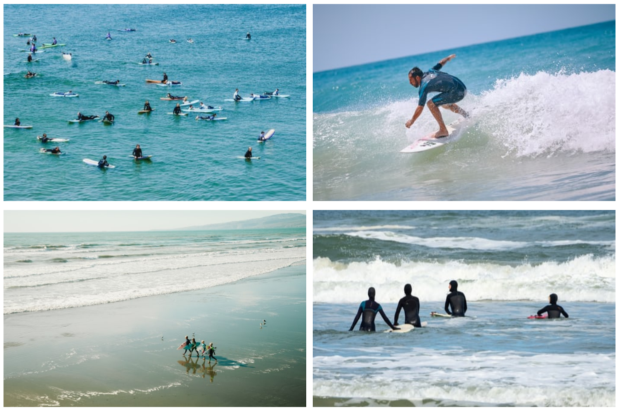


Figure 1: Several images with the same label set: *person*, *surfboard* and *sea*. Given this simple label set, we observe that a large and diverse set of layouts is plausible.

1. Introduction

Scene generation, which usually consists of realistic generation of multiple objects under a semantic layout, remains one of the core frontiers of computer vision. Despite the rapid progress and recent successes in object generation (*e.g.*, celebrity face, animals, *etc.*) [1, 9, 13] and scene generation [4, 11, 12, 19, 22, 30, 31], little attention has been paid to frameworks designed for stochastic semantic layout generation. Having a robust model for layout generation will not only allow us to generate reliable scene layouts, but also provide priors and means to infer latent relationships between objects, advancing progress in the scene understanding domain.

A plausible semantic layout calls for reasonable *spatial* and *count* relationships (relationships between the number of instances of different labels) between objects in a scene [5, 27]. For example, a person would either ride (*on top of*) a horse, or stand *next to* a horse, but seldom would he be *under* a horse. Another example would be that the number of ties would very likely be smaller than or equal to the number of people in an image. The first example shows an instance of a plausible spatial relationship and the second shows a case of a generic count relationship. Such intrinsic relationships buried in high-dimensional visual data are usually learned implicitly by mapping the textual description to visual data. However, since the textual description can always be treated as an abstraction of the visual data, the process becomes a *one-to-many* mapping. In other words, given the same text information as a condition, a good model should be able to generate multiple plausible images all of which satisfy the semantic description.

Previous works focused on a popular simplified instance of the problem described above: scene generation based on sentence description [6, 11, 12, 19, 24, 25, 30]. A typical sentence description includes partial information on both the background and objects, along with details of the objects' appearances and scene layout. These frameworks rely heavily on the extra relational information provided by the sentence. As a result, although these methods manage to generate realistic scenes, they tend to ignore learning the intrinsic relationships between the objects, prohibiting the wide adoption of such models where weaker descriptions are provided.

In this work, we consider a more sophisticated problem: scene generation based on a label set description. A label set, as a much weaker description, only provides the set of labels present in the image (without any additional relationship description), requiring the model to learn *spatial* and *count* relationships from visual data.

Furthermore, the ambiguity of this type of label set supervision calls for diverse scene generation. For example, given the label set *person, surfboard, sea*, a corresponding scene could have multiple instances of each label (under plausible count relationships), positioned at various locations (under plausible spatial relationships). For instance in the COCO dataset [21], there are 869 images in the training set that have the label set *person*, *sea* and *surfboard*. Figure 1 shows examples of multiple plausible images with this label set.

We propose LayoutVAE, a stochastic model capable of generating scene layouts given a label set. The proposed framework can be easily embedded into existing scene generation models that take scene layout as input, such as [10, 31], providing them plausible and diverse layouts. Our main contributions are as follows.

- We propose a new model for stochastic scene layout generation given a label set input. Our model has two components, one to model the distributions of count relationships between objects and another to model the distributions of spatial relationships between objects.
- We propose a new synthetic dataset, MNIST-Layouts, that captures the stochastic nature of scene layout generation problem.
- We experimentally validate our model using MNIST-Layouts and the COCO [21] dataset which contains complex real world scene layouts. We analyze our model and show that it can be used to detect unlikely scene layouts.

2. Related Work

Sentence-conditioned image generation. A variety of models have proposed to generate an image given a sentence. Reed *et al.* [25] use a GAN [7] that is conditioned on a text encoding for generating images. Zhang *et al.* [30] propose a GAN based image generation framework where the image is progressively generated in two stages at increasing resolutions. Reed *et al.* [24] perform image generation with sentence input along with additional information in the form of keypoints or bounding boxes.

Hong *et al.* [11] break down the process of generating an image from a sentence into multiple stages. The input sentence is first used to predict the objects that are present in the scene, followed by prediction of bounding boxes, then semantic segmentation masks, and finally the image. While scene layout generation in this work predicts probability distributions for bounding box layout, it fails to model the stochasticity intrinsic in predicting each bounding box. Gupta *et al.* [8] use an approach similar to [11] to predict layouts for generating videos from scripts. Johnson *et al.* [12] uses the scene graph generated from the input sentence as input to the image generation model. Given a scene graph, their model can generate only one scene layout.

Deng *et al.* [6] propose PNP-Net, a VAE framework to generate image of an abstract scene from a text based program that fully describes it. While PNP-Net is a stochastic model for generation, it was tested on synthetic datasets with only a small number of classes. Furthermore, it tries to encode the entire image into a single latent code whereas, in LayoutVAE, we break down just the layout generation step into two stages with multiple steps in each stage. Based on these reasons, it is unclear whether PNP-Net can scale up to real world image datasets with a large number of classes. Tao *et al.* [29] propose a GAN based model with attention for sentence to image generation. The more recent work from Li *et al.* [19] follow a multi-stage approach similar to [11] to generate an image from a sentence, with the key novelty of using attention mechanisms to create more realistic objects in the image.

Layout generation in other contexts. Chang *et al.* [3] propose a method for 3D indoor scene generation based on text description by placing objects from a 3D object library, and is later improved in [2] by learning the grounding of more detailed text descriptions with 3D objects. Wang *et al.* [28] use a convolutional network that iteratively generates a 3D room scene by adding one object at a time. Qi *et al.* [23] propose a spatial And-Or graph to represent indoor scenes, from which new scenes can be sampled. Different from most other works, they use human affordances and activity information with respect to objects in the scene to model probable spatial layouts. Li *et al.* [18] propose a VAE based framework that encodes object and layout information of indoor 3D scenes in a latent code. During generation, the latent code is recursively decoded to obtain details of individual objects and their layout.

More recently, Li *et al.* [17] proposed LayoutGAN, a GAN based model that generates layouts of graphic elements (rectangles, triangles etc.). While this work is close to ours in terms of the problem focus, LayoutGAN generates label sets based on input noise, and it cannot generate layout for a given set of labels.

Placing objects in scenes. Lee *et al.* [16] propose a conditional GAN model for the problem of adding the segmentation mask of a new object to the semantic segmentation of an image. Lin *et al.* [20] address a similar problem of adding an RGB mask of an object into a background image.

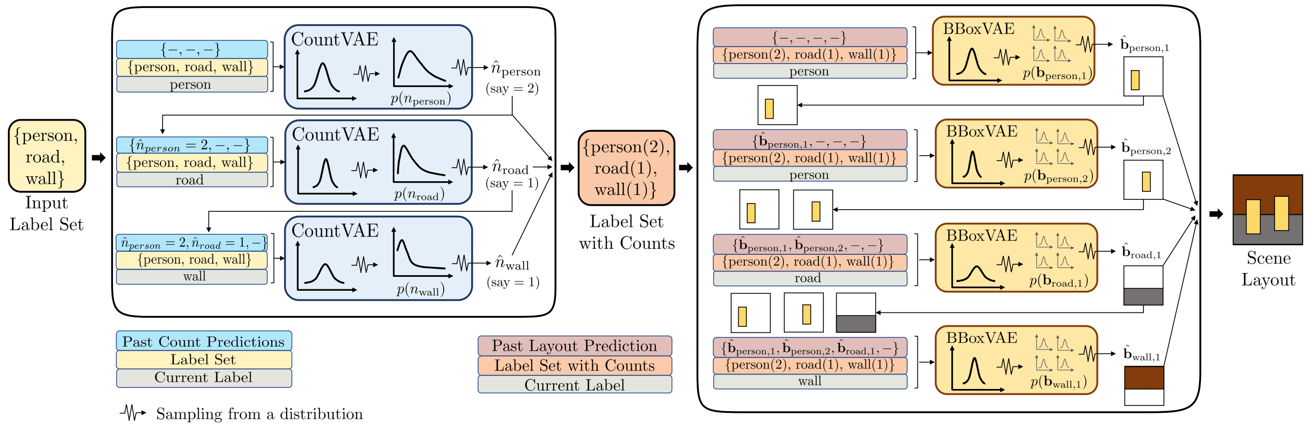

{person, road, wall}
Input Label Set
CountVAE
{−, −, −}
{person, road, wall}
person
$p(n_{\text{person}})$
$\hat{n}_{\text{person}}$ (say = 2)
{$\hat{n}_{person} = 2, −, −$}
{person, road, wall}
road
$p(n_{\text{road}})$
$\hat{n}_{\text{road}}$ (say = 1)
$\hat{n}_{person} = 2, \hat{n}_{road} = 1, −$}
{person, road, wall}
wall
$p(n_{\text{wall}})$
$\hat{n}_{\text{wall}}$ (say = 1)
{person(2), road(1), wall(1)}
Label Set with Counts
BBoxVAE
{−, −, −, −}
{person(2), road(1), wall(1)}
person
$p(\mathbf{b}_{\text{person},1})$
$\hat{\mathbf{b}}_{\text{person},1}$
{$\hat{\mathbf{b}}_{\text{person},1}$, −, −, −}
$p(\mathbf{b}_{\text{person},2})$
$\hat{\mathbf{b}}_{\text{person},2}$
{$\hat{\mathbf{b}}_{\text{person},1}, \hat{\mathbf{b}}_{\text{person},2}$, −, −}
road
$p(\mathbf{b}_{\text{road},1})$
$\hat{\mathbf{b}}_{\text{road},1}$
{$\hat{\mathbf{b}}_{\text{person},1}, \hat{\mathbf{b}}_{\text{person},2}, \hat{\mathbf{b}}_{\text{road},1}$, −}
wall
$p(\mathbf{b}_{\text{wall},1})$
$\hat{\mathbf{b}}_{\text{wall},1}$
Scene Layout
Past Count Predictions
Label Set
Current Label
Past Layout Prediction
Label Set with Counts
Current Label
Sampling from a distribution

Figure 2: **Model**. LayoutVAE is composed of two models: CountVAE which predicts the number of objects for each category and BBoxVAE which predicts the bounding box of each object. A graphical model is given in **??** of the supplementary.

3. Background

In this section, we first define the problem of scene layout generation from a label set, and then provide an overview of the base models that LayoutVAE is built upon.

3.1. Problem Setup

We are interested in modeling contextual relationships between objects in scenes, and furthermore generation of diverse yet plausible scene layouts given a label set as input. The problem can be formulated as follows.

Given a collection of M object categories, we represent the label set corresponding to an image in the dataset as $\mathbb{L} \subseteq \{1, 2, 3, ..., M\}$ which indicates the categories present in the image. Note that here we use the word "object" in its very general form: *car*, *cat*, *person*, *sky* and *water* are objects. For each label $k \in \mathbb{L}$, let n_k be the number of objects of that label in the image and $\mathbb{B}_k = \{\mathbf{b}_{k,1}, \mathbf{b}_{k,2}, ..., \mathbf{b}_{k,n_k}\}$ be the set of bounding boxes. $\mathbf{b}_{k,i} = [x_{k,i}, y_{k,i}, w_{k,i}, h_{k,i}]$ represents the top-left coordinates, width and height of the i-th bounding box of category k. We train a generative model to predict diverse yet plausible sets of $\{\mathbb{B}_k : k \in \mathbb{L}\}$ given the label set $\mathbb{L}$ as input.

3.2. Base Models

Variational Autoencoders. A variational autoencoder (VAE) [15] describes an instance of a family of generative models $p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})$ with a complex likelihood function $p_\theta(\mathbf{x}|\mathbf{z})$ and an amortized inference network $q_\phi(\mathbf{z}|\mathbf{x})$ to approximate the true posterior $p_\theta(\mathbf{z}|\mathbf{x})$. Here $\mathbf{x}$ represents observable data examples, $\mathbf{z}$ the latent codes, θ the generative model parameters, and ϕ the inference network parameters. To prevent the latent variable $\mathbf{z}$ from just copying $\mathbf{x}$, we force $q_\phi(\mathbf{z}|\mathbf{x})$ to be close to the prior distribution $p_\theta(\mathbf{z})$ using a KL-divergence term. Usually in VAE models, $p_\theta(\mathbf{z})$ is a fixed Gaussian $\mathcal{N}(0, I)$. Both the generative and the inference networks are realized as non-linear neural networks. An evidence lower bound (ELBO) $\mathcal{L}(\mathbf{x}; \theta, \phi)$ on the generative data likelihood $\log p(\mathbf{x})$ is used to jointly optimize θ and ϕ:

$$\mathcal{L}(\mathbf{x}; \theta, \phi) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \text{KL}\left(q_\phi(\mathbf{z}|\mathbf{x}) || p_\theta(\mathbf{z})\right) \tag{1}$$

Conditional VAEs. A conditional VAE (CVAE) [26] defines an extension of VAE that conditions on an auxiliary description $\mathbf{c}$ of the data. The auxiliary conditional variable makes it possible to infer a conditional posterior $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{c})$ as well as perform generation $p_\theta(\mathbf{x}|\mathbf{z}, \mathbf{c})$ based on a given description $\mathbf{c}$. The ELBO is thus updated as:

$$\begin{aligned} \mathcal{L}^{CVAE}(\mathbf{x}, \mathbf{c}; \theta, \phi) =& \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{c})} \left[\log p_\theta(\mathbf{x}|\mathbf{z}, \mathbf{c})\right] \\ & - \text{KL}\left(q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{c}) || p_\theta(\mathbf{z}|\mathbf{c})\right) \end{aligned} \tag{2}$$

In CVAE models, the prior of the latent variables $\mathbf{z}$ is modulated by the auxiliary input $\mathbf{c}$.

4. LayoutVAE for Stochastic Scene Layout Generation

In this section, we present LayoutVAE and describe the scene layout generation process given a label set. As discussed in section 1, this task is challenging and solving it requires answering the following two questions: *what is the number of objects for each category?* and *what is the location and size for each object?* LayoutVAE is naturally decomposed into two models: one to predict the count for each given label, named *CountVAE*, and another to predict the location and size of each object, named *BBoxVAE*. The overall structure of the proposed LayoutVAE is shown in Figure 2. The number of objects (count) for each label is first predicted by CountVAE, then BBoxVAE predicts the bounding box for each object. The two-step approach with stochastic models naturally allows LayoutVAE to generate diverse layouts. In addition, it provides the flexibility to handle various types of input as it allows us to use each module independently. For example, BBoxVAE can be used to generate a layout if counts are available, or add a single bounding box in an existing image given a new label.

The input to CountVAE is the set of labels $\mathbb{L}$ and it predicts the distributions of object counts $\{n_k : k \in \mathbb{L}\}$ autoregressively, where n_k is the object count for category k. The input of BBoxVAE is the set of labels along with the counts for each of the label $\{n_k : k \in \mathbb{L}\}$ and it predicts the distribution of each bounding box $\mathbf{b}_{k,i}$ autoregressively.

4.1. CountVAE

CountVAE is an instance of conditional VAE designed to predict conditional count distribution for the labels in an autoregressive manner. We use a predefined order for the label set (we observe empirically that a predefined order is superior to randomized order across samples; learning an order is a potential extension but adds complexity). In practice, CountVAE predicts the count of the first label, then the count of the second label *etc*., at each step conditioned on already predicted counts. It models the distribution of count n_k given the label set $\mathbb{L}$, the current label k and the counts for each category that was predicted before $\{n_m : m < k\}$. The conditioning input for CountVAE is:

$$\mathbf{c}_k^c = (\mathbb{L}, k, \{n_m : m < k\}) \tag{3}$$

where $(\cdot, \cdot)$ denotes a tuple. We use the notation of superscript c to indicate that it is related to the CountVAE. We use a Poisson distribution to model the number of occurrences of the current label at each step:

$$p_{\theta^c}(n_k|\mathbf{z}_k^c, \mathbf{c}_k^c) = \frac{(\lambda(\mathbf{z}_k^c, \mathbf{c}_k^c))^{(n_k-1)} e^{-\lambda(\mathbf{z}_k^c, \mathbf{c}_k^c)}}{(n_k - 1)!} \tag{4}$$

where $p_{\theta^c}(n_k|\mathbf{z}_k^c, \mathbf{c}_k^c)$ is a probability distribution over random variable n_k, θ^c represents the generative model parameters of CountVAE, and $\lambda(\mathbf{z}_k^c, \mathbf{c}_k^c)$ is the rate which depends on the latent variable sample $\mathbf{z}_k^c$ and the conditional input $\mathbf{c}_k^c$. Note that we learn the distribution over $n_k - 1$ since the count per label is always 1 or greater in this problem setting.

The latent variable is sampled from the approximate posterior during learning and from the conditional prior during generation. The latent variables model the ambiguity in the scene layout. Both approximate posterior and prior are modeled as multivariate Gaussian distributions with diagonal covariance with parameters as shown below:

$$q_{\phi^c}(\mathbf{z}_k^c|n_k, \mathbf{c}_k^c) = \mathcal{N}(\mu_{\phi^c}(n_k, \mathbf{c}_k^c), \sigma^2_{\phi^c}(n_k, \mathbf{c}_k^c)) \tag{5}$$

$$p_{\theta^c}(\mathbf{z}_k^c|\mathbf{c}_k^c) = \mathcal{N}(\mu_{\theta^c}(\mathbf{c}_k^c), \sigma^2_{\theta^c}(\mathbf{c}_k^c)) \tag{6}$$

Algorithm 1 CountVAE: Loss computation on the image

```
Input: Label set $\mathbb{L}$, instance count $\{n_k : k \in \mathbb{L}\}$
$\mathcal{L}^{Count} = 0$
for $k \in \mathbb{L}$ do
    Compute the conditioning input $\mathbf{c}_k^c$ for category $k$ (Equation 3)
    Compute the variational lower bound $\mathcal{L}^c(n_k, \mathbf{c}_k^c; \theta^c, \phi^c)$ (Equation 7)
    $\mathcal{L}^{Count} = \mathcal{L}^{Count} + \mathcal{L}^c(n_k, \mathbf{c}_k^c; \theta^c, \phi^c)$
end for
Output: $\mathcal{L}^{Count}/|\mathbb{L}|$
```

where μ_{ϕ^c} (resp. μ_{θ^c}) and $\sigma^2_{\phi^c}$ (resp. $\sigma^2_{\theta^c}$) are functions that estimate the mean and the variance of the approximate posterior (resp. prior). Details of the model architecture are given in **??** of the supplementary.

Learning. The model is optimized by maximizing the ELBO over $\{n_k : k \in \mathbb{L}\}$. The ELBO corresponding to the label count n_k is given by

$$\begin{aligned}\mathcal{L}^c(n_k, \mathbf{c}_k^c; \theta^c, \phi^c) =& \mathbb{E}_{q_{\phi^c}(\mathbf{z}_k^c|n_k, \mathbf{c}_k^c)}\left[\log p_{\theta^c}(n_k|\mathbf{z}_k^c, \mathbf{c}_k^c)\right] \\ &- \mathrm{KL}\left(q_{\phi^c}(\mathbf{z}_k^c|n_k, \mathbf{c}_k^c)||p_{\theta^c}(\mathbf{z}_k^c|\mathbf{c}_k^c)\right)\end{aligned} \tag{7}$$

where ϕ^c represents the inference model parameters of CountVAE. Log-likelihood of the ground truth count under the predicted Poisson distribution is used to compute $p_{\theta^c}(n_k|\mathbf{z}_k^c, \mathbf{c}_k^c)$, while the KL divergence between the two Gaussian distributions is computed analytically. The computation of the loss for the label set $\mathbb{L}$ is given in Algorithm 1.

Generation. Given a label set $\mathbb{L}$, the CountVAE autoregressively predicts the object count for each category by sampling from the count distribution (Equation 4). We now present the generation process to predict the object count of category k. We first compute the conditional input $\bar{\mathbf{c}}_k^c$:

$$\bar{\mathbf{c}}_k^c = (\mathbb{L}, k, \{\hat{n}_m : m < k\}) \tag{8}$$

where $\hat{n}_m$ is the predicted instance count for category m. Note that to predict the instance count of category k, the model exploits the predicted counts $\hat{n}_m$ for previous categories $m < k$ to get more consistent counts. Then, we sample a latent variable $\hat{\mathbf{z}}_k^c$ from the conditional prior:

$$\hat{\mathbf{z}}_k^c \sim p_{\theta^c}(\mathbf{z}_k^c|\bar{\mathbf{c}}_k^c) \tag{9}$$

Finally, the count is sampled from the predicted Poisson count distribution:

$$\hat{n}_k \sim p_{\theta^c}(n_k|\hat{\mathbf{z}}_k^c, \bar{\mathbf{c}}_k^c) \tag{10}$$

This label count is further used in the conditioning variable for the future steps of CountVAE.

Algorithm 2 BBoxVAE: Loss computation on the image

```
Input: Label set $\mathbb{L}$, instance count $\{n_k : k \in \mathbb{L}\}$, set of
    bounding boxes $\{\mathbb{B}_k : k \in \mathbb{L}\}$
$\mathcal{L}^{BBox} = 0$
for $k \in \mathbb{L}$ do
    for $j \in \{1, \dots, n_k\}$ do
        Compute the conditioning input $\mathbf{c}^b_{k,j}$ for the $j$-th
            bounding box of category $k$ (Equation 11)
        Compute the variational lower bound
            $\mathcal{L}^b(\mathbf{b}_{k,j}, \mathbf{c}^b_{k,j}; \theta^b, \phi^b)$ (Equation 13)
        $\mathcal{L}^{BBox} = \mathcal{L}^{BBox} + \mathcal{L}^b(\mathbf{b}_{k,j}, \mathbf{c}^b_{k,j}; \theta^b, \phi^b)$
    end for
end for
Output: $\mathcal{L}^{BBox} / \left(\sum_{k \in \mathbb{L}} n_k\right)$
```

4.2. BBoxVAE

Given the label set $\mathbb{L}$ and the object count per category $\{n_m : m \in \mathbb{L}\}$, the BBoxVAE predicts the distribution of coordinates for the bounding boxes autoregressively. We follow the same predefined label order as CountVAE in the label space, and order the bounding boxes from left to right for each label. All bounding boxes for a given label are predicted before moving on to the next label. BBoxVAE is a conditional VAE that models the j^{th} bounding box $\mathbf{b}_{k,j}$ for the label k given the label set $\mathbb{L}$ along with the count for each label, current label k and coordinate and label information of all the bounding boxes that were predicted earlier. Previous predictions include all the bounding boxes for previous labels as well as the ones previously predicted for the current label: $\mathbb{B}^{prev}_{k,j} = \{\mathbf{b}_{m,:} : m < k\} \cup \{\mathbf{b}_{k,i} : i < j\}$ The conditioning input of the BBoxVAE is:

$$\mathbf{c}^b_{k,j} = \left(\{n_m : m \in \mathbb{L}\}, k, \mathbb{B}^{prev}_{k,j}\right) \tag{11}$$

We use the notation of exponent b to indicate that it is related to the BBoxVAE. We model bounding box coordinates and size information using a quadrivariate Gaussian distribution as shown in Equation 12. We omit the subscript k, j for all variables in the equation for brevity.

$$p_{\theta^b}(x, y, w, h|\mathbf{z}^b, \mathbf{c}^b) = \mathcal{N}\left(x, y, w, h|\mu^b(\mathbf{z}^b, \mathbf{c}^b), \sigma^b(\mathbf{z}^b, \mathbf{c}^b)\right) \tag{12}$$

where θ^b represents the generative model parameters of BBoxVAE, $\mathbf{z}^b$ denotes a sampled latent variable and μ^b and σ^b are functions that estimate the mean and the variance respectively for the Gaussian distribution. x (resp. y) is the x- (resp. y-) coordinate of the top left corner and w (resp. h) is the width (resp. height) of the bounding box. These variables are normalized between 0 and 1 to be independent of the dimensions of the image. Details of the model architecture are given in **??** of the supplementary.

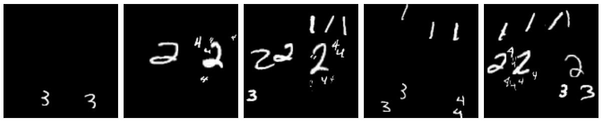


Figure 3: **Samples from MNIST-Layouts dataset.** We randomly sample MNIST digits of appropriate labels to fill in the bounding boxes. The rules for count and spatial layout (e.g. large 2s in the middle, small 3s at the bottom etc.) are described in **??** in the supplementary.

Learning. We train the model in an analogous manner to CountVAE by maximizing the variational lower bound over the entire set of bounding boxes. For the bounding box $\mathbf{b}_{k,j}$, the lower bound is given by (omitting the subscript k, j for all variables in the equation):

$$\begin{aligned}\mathcal{L}^b(\mathbf{b}, \mathbf{c}^b; \theta^b, \phi^b) =& \mathbb{E}_{q_{\phi^b}(\mathbf{z}^b|\mathbf{b}, \mathbf{c}^b)}\left[\log p_{\theta^b}(\mathbf{b}|\mathbf{z}^b, \mathbf{c}^b)\right] \\ &- \mathrm{KL}\left(q_{\phi^b}(\mathbf{z}^b|\mathbf{b}, \mathbf{c}^b)||p_{\theta^b}(\mathbf{z}^b|\mathbf{c}^b)\right)\end{aligned} \tag{13}$$

where ϕ^b represents the inference model parameters of BBoxVAE. The computation of the loss for all the bounding boxes is given in Algorithm 2.

Generation. Generation for BBoxVAE is performed in an analogous manner as CountVAE. Given a label set along with the number of instances of each label, BBoxVAE autoregressively predicts bounding box distributions (Equation 12) by sampling a latent variable from the conditional prior $p_{\theta^b}(\mathbf{z}^b|\mathbf{c}^b)$. The conditioning variable is updated after each step by including a sampled bounding box from the current step prediction.

5. Experiments

We implemented LayoutVAE in PyTorch, and used Adam optimizer [14] for training the models. CountVAE was trained for 50 epochs at a learning rate of 10^{-5} with batch size 32. BBoxVAE was trained for 150 epochs at a learning rate of 10^{-4} with batch size 32. We chose the best model during training based on the validation loss, and evaluate that model's performance on the test set. We present quantitative and qualitative results that showcase the usefulness of LayoutVAE model.

5.1. Datasets

MNIST-Layouts. We created this dataset by placing multiple MNIST digits on a 128×128 canvas based on predefined rules detailed in **??** in the supplementary material. We use the digits $\{1, 2, 3, 4\}$ as the global set of labels, thus limiting the maximum number of labels per image to 4. The dataset consists of 5000 training images, and 1000 images each for validation and testing. Figure 3 shows some examples of scene layouts from the training set.

Model	Label Count		BBox
	Accuracy (%)	Accuracy within ±1 (%)	Mean IoU
AR-MLP	74.21	87.50	0.18
BBoxLSTM	-	-	0.17
BLSTM	73.98	87.87	0.15
sg2im [12]	-	-	0.14
LayoutVAE	**78.38**	**89.87**	**0.20**

Table 1: **Comparison with baselines on COCO dataset using accuracy metrics.** We use the most likely count or bounding box as the prediction for all the models. While all models predict a distribution as output, only LayoutVAE has a stochastic latent code. Mean of the latent code distribution is used to obtain the output (count or bounding box) distribution for measuring the accuracy of LayoutVAE.

COCO. We use 2017 Panoptic version of COCO dataset [21] for our experiments. This dataset has images from complex everyday scenes with bounding box annotations for all major *things* and *stuffs* present in the scene. The official release has $118,267$ annotated images in the training set and 5000 in the validation set. To allow a comparison with future methods, we use the official validation set as test set, and use the last 5000 images from the training set as validation set in our experiments. This dataset has 80 *thing* categories (person, table, chair *etc*.) and 53 *stuff* categories (sky, sand, snow *etc*.). The largest number of labels present in a label set is 34, and the largest number of bounding box instances present in an image is 98. We normalize all bounding box dimensions to $[0, 1]$ by dividing by the image size (width or height, as appropriate). This allows the model to predict layouts for square images of any size. We ignore *thing* instances that are tagged *"iscrowd" i.e.* consists of a group of thing instances. They account for less than 1 percent of all the annotations.

5.2. Baseline Models

To our knowledge, the problem of generating scene layouts (with both *thing* and *stuff* categories, and multiple bounding boxes per category covering almost the entire image) from a label set is new and has not been studied. There is no existing model for this task so we adapt existing models that generate scene layout from more complex inputs *e.g*. sentence, scene graphs. Note that we choose the same distributions as in LayoutVAE for modeling the outputs (counts or bounding box information) to have a fair comparison between all the models.

Autoregressive MLP (AR-MLP). This model is analogous to our proposed VAE based model, except that it has a multilayer perceptron instead of a VAE. It runs autoregressively and takes as input the same conditioning information used in the VAE models. As in LayoutVAE, we have two separate sub-models – *CountMLP* for predicting count distribution given the corresponding conditioning information, and *BBoxMLP* for predicting bounding box distribution given the corresponding conditioning information.

Model	Label Count		BBox	
	MNIST	COCO	MNIST	COCO
AR-MLP	1.246	0.740	5.83	40.91
BBoxLSTM	-	-	6.21	42.42
BLSTM	1.246	0.732	20.06	52.84
sg2im [12]	-	-	-	214.03
LayoutVAE	1.246	**0.562**	**−0.07**	**2.72**

Table 2: **Comparison with baselines using likelihood metric.** Negative log-likelihood (lower is better) over the test set on MNIST-Layouts and COCO datasets.

BBoxLSTM. This model is analogous to the bounding box predictor used in Hong *et al*. [11]. It consists of an LSTM that takes in the label set embedding (as opposed to sentence embedding in [11]) at each step and predicts the label and bounding box distributions for bounding boxes in the image one by one. At each step, the LSTM output is first used to generate the label distribution. The bounding box distribution is then generated conditioned on the label.

BLSTM. We also use bidirectional LSTM in an analogous fashion as LayoutVAE. We have *CountBLSTM* and *BBoxBLSTM* to predict count distribution and bounding box distribution respectively, where we input the respective conditioning information at each step for the BLSTM. The conditioning information is similar as in the VAE models except that we do not give the pooled embedding of previous counts/bounding boxes information as we now have a bidirectional recurrent model. Note that having a bidirectional model can possibly alleviate the need to explicitly define the order in the label and the bounding box coordinates spaces.

sg2im [12]. This model generates a scene layout based on a scene graph. We compare LayoutVAE with this model because label set (in this case, with multiple copies of labels as per the ground truth count of each label) can be seen as the simplest scene graph without any relations. We also explored another strategy where a scene graph is randomly generated for the label set, but we found that model performance worsened in that setting. For this experiment, we use the code and the pretrained model on COCO provided by the authors at `https://github.com/google/sg2im` to predict the bounding boxes.

Note that these models are limited in their ability to model complex distributions and generate diverse layouts, whereas LayoutVAE can do so using the stochastic latent variable.

History	Context	NLL	
		CountVAE	BBoxVAE
		0.592	4.17
	✓	0.570	3.78
✓		0.581	2.93
✓	✓	**0.562**	**2.72**

Table 3: **Ablation study.** Test NLL results for CountVAE and BBoxVAE by prior sampling on COCO dataset. The history is the previous counts for CountVAE and the previous bounding box information for BBoxVAE. The context is the label set for CountVAE and the label set with counts for BBoxVAE.

things before *stuffs*	*stuffs* before *things*	random
2.72	**2.71**	3.22

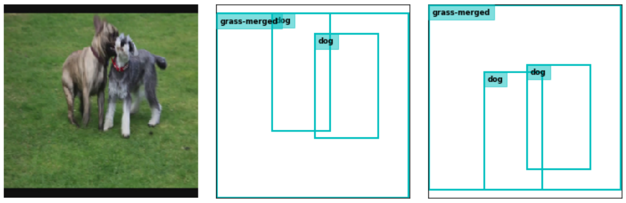
Table 4: **Analysis of label order.** NLL for BBoxVAE for different label set order on COCO dataset.

5.3. Quantitative Evaluation

To compare the models, we compute average negative log-likelihood (NLL) of the test samples' count or bounding box coordinates under the respective predicted distributions. We compute Monte Carlo estimate of NLL for the VAE models by drawing 1000 samples from the conditional prior at each step of generation. LayoutVAE and the baselines are trained and evaluated by teacher forcing *i.e.* we provide the ground truth context and count/bounding box history at each step of computation.

Comparison with baseline models. Table 1(accuracy metrics) and Table 2(likelihood) show count and bounding box generation performances for all the models. First, we observe that LayoutVAE model is significantly better than all the baselines. In particular the large improvement with respect to the autoregrssive MLP baseline shows the relevance of stochastic model for this task. It is also interesting to note that autoregressive MLP model performs better that the recurrent models. Finally, we observe that sg2im model [12] is not able to predict accurate bounding boxes without a scene graph with relationships. LayoutVAE and the baselines show similar performance for count prediction in MNIST-Layouts because estimating count distribution over the 4 labels of MNIST-Layouts is much easier than in the real world data from COCO.

Ablation study. In Table 3, we analyze the importance of context representation and history in the conditioning information. We observe that both history and context are useful because they increase the log-likelihood but their effects vary for count and bounding box models. The context is more critical than the history for the CountVAE whereas the history is more critical than the context for the BBoxVAE. Despite these different behaviours, we note that history and context are complementary and increase the performances for both CountVAE and BBoxVAE. This experiment validates the importance of both context and history in the conditioning information for both CountVAE and BBoxVAE.



original image original layout flipped layout

Figure 4: Example where likelihood under LayoutVAE increases when flipped upside down. We can see that the flipped layout is equally plausible for this example.

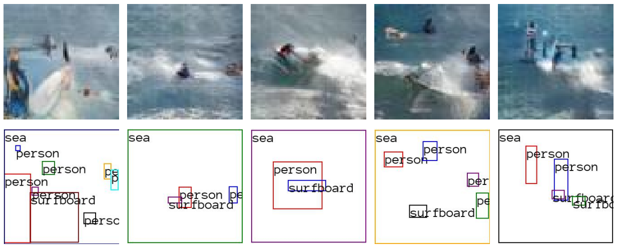
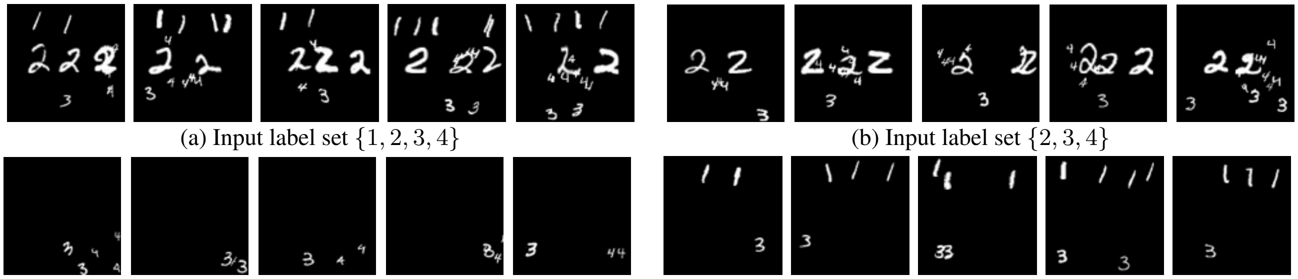


Figure 5: **Image Generation from a label set.** We show diverse realistic layouts generated by LayoutVAE for the input label set *person*, *surfboard* and *sea*. We generate images from the layouts by using the generative model provided by Zhao *et al*. [31]

Analysis of the label set order. We performed experiments by changing the order of labels in the label set. We consider three variants for this experiments — a fixed order with all the *things* before *stuffs* (default choice for all the other experiments), reverse order of the above with all the *stuffs* before *things*, and finally we randomly assigned the order of labels for each image. Table 4 shows the results of the experiment on BBoxVAE. We found that fixed order (*things* first or *stuffs* first) gave similar results, while randomizing the order of labels across images resulted in a significant reduction in performance.

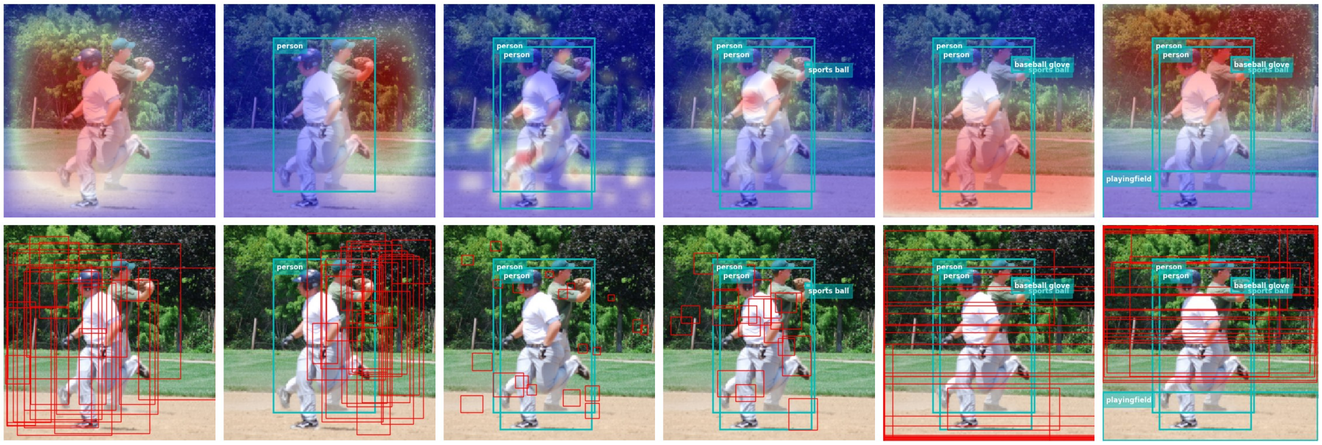
Detecting unlikely layouts in COCO. To test the ability of the model to differentiate between plausible and unlikely layouts, we perform an experiment where we flip the image layouts in the test set upside down, and evaluate the NLL (in this case, by importance sampling using 1000 samples from the posterior) of both the normal layout and the flipped layout under BBoxVAE. We found that the NLL got worse for 92.58% of image layouts in the test set when flipped upside down. The average NLL for original layouts is 2.26 while that for the flipped layouts is 4.33. We note that there are instances where flipping does not necessarily lead to an



(a) Input label set $\{1, 2, 3, 4\}$ (b) Input label set $\{2, 3, 4\}$

(c) Input label set $\{3, 4\}$ (d) Input label set $\{1, 3\}$

Figure 6: **Stochastic layout generation for MNIST-Layouts.** We show 5 sample layouts generated by LayoutVAE for each input set of labels. We fill in the generated bounding boxes with randomly sampled MNIST digits of appropriate labels. We can see that LayoutVAE generates bounding boxes following all the rules that we defined for MNIST-Layouts (e.g. large 2s in the middle, small 3s at the bottom etc. with the appropriate count values). The complete set of rules is given in **??** in the supplementary material.



person person sports ball baseball glove playingfield tree-merged

Figure 7: **Bounding box prediction using LayoutVAE.** We show the steps of bounding box generation for a test set sample. At each step, we obtain diverse bounding box predictions for the input label (written in subcaptions) given the ground truth layout up to the previous step (shown in cyan bounding boxes). We show bounding box heatmap using 100 samples from the prior distribution (top row for each step, red means high probability), and bounding boxes using 20 samples from the prior distribution. The predicted layouts are overlaid on the test image for clarity.

anomalous layout, which explains why likelihood worsened only for 92.58% of the test set layouts. Figure 4 shows such an example where flipping led to an increased likelihood under the model. Additional results are shown in **??** in the supplementary material.

5.4. Qualitative Evaluation

Scene layout samples. We present qualitative results for diverse layout generation on COCO(Figure 5) datasets and MNIST-Layouts(Figure 6). The advantages of modeling scene layout generation using a probabilistic model is evident from these results: LayoutVAE learns the rules of object layout in the scene and is capable of generating diverse layouts with different counts of objects. More examples can be found in **??** in the supplementary material.

Per step bounding box prediction for COCO. Figure 7 shows steps of bounding box generation for a test set of labels. For each step, we show the probability map and 20 plausible bounding boxes. We observe that given some bounding boxes, the model is able to predict where the next object could be in the image. More examples for bounding box generation are provided in **??** in the supplementary material.

6. Conclusion

We propose LayoutVAE for generating stochastic scene layouts given a label set as input. It comprises of two components to model the distribution of spatial and count relationships between objects in a scene. We compared it with other existing methods or analogues thereof, and showed significant performance improvement. Qualitative visualizations show that LayoutVAE can learn intrinsic relationships between objects in real world scenes. Furthermore, LayoutVAE can be used to detect abnormal layouts.

References

[1] Navaneeth Bodla, Gang Hua, and Rama Chellappa. Semi-supervised FusedGAN for Conditional Image Generation. In *European Conference on Computer Vision (ECCV)*, 2018. 1

[2] Angel Chang, Will Monroe, Manolis Savva, Christopher Potts, and Christopher D. Manning. Text to 3D Scene Generation with Rich Lexical Grounding. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, 2015. 2

[3] Angel X Chang, Manolis Savva, and Christopher D Manning. Learning Spatial Knowledge for Text to 3D Scene Generation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014. 2

[4] Qifeng Chen and Vladlen Koltun. Photographic Image Synthesis With Cascaded Refinement Networks. In *IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 1

[5] Myung Jin Choi, Joseph J Lim, Antonio Torralba, and Alan S Willsky. Exploiting hierarchical context on a large database of object categories. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010. 1

[6] Zhiwei Deng, Jiacheng Chen, Yifang Fu, and Greg Mori. Probabilistic Neural Programmed Networks for Scene Generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 1, 2

[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems (NeurIPS)*. 2014. 2

[8] Tanmay Gupta, Dustin Schwenk, Ali Farhadi, Derek Hoiem, and Aniruddha Kembhavi. Imagine This! Scripts to Compositions to Videos. In *European Conference on Computer Vision (ECCV)*, 2018. 2

[9] Yang He, Bernt Schiele, and Mario Fritz. Diverse Conditional Image Generation by Stochastic Regression with Latent Drop-Out Codes. In *European Conference on Computer Vision (ECCV)*, 2018. 1

[10] Tobias Hinz, Stefan Heinrich, and Stefan Wermter. Generating Multiple Objects at Spatially Distinct Locations. In *International Conference on Learning Representations (ICLR)*, 2019. 2

[11] Seunghoon Hong, Dingdong Yang, Jongwook Choi, and Honglak Lee. Inferring Semantic Layout for Hierarchical Text-to-Image Synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2, 6

[12] Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image Generation from Scene Graphs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2, 6, 7

[13] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive Growing of GANs for Improved Quality, Stability, and Variation. In *International Conference on Learning Representations (ICLR)*, 2018. 1

[14] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*, 2015. 5

[15] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. *International Conference on Learning Representations (ICLR)*, 2014. 3

[16] Donghoon Lee, Ming-Yu Liu, Ming-Hsuan Yang, Sifei Liu, Jinwei Gu, and Jan Kautz. Context-Aware Synthesis and Placement of Object Instances. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 2

[17] Jianan Li, Tingfa Xu, Jianming Zhang, Aaron Hertzmann, and Jimei Yang. LayoutGAN: Generating Graphic Layouts with Wireframe Discriminator. In *International Conference on Learning Representations (ICLR)*, 2019. 2

[18] Manyi Li, Akshay Gadi Patil, Kai Xu, Siddhartha Chaudhuri, Owais Khan, Ariel Shamir, Changhe Tu, Baoquan Chen, Daniel Cohen-Or, and Hao Zhang. GRAINS: Generative Recursive Autoencoders for Indoor Scenes. *ACM Transactions on Graphics (TOG)*, 2018. 2

[19] Wenbo Li, Pengchuan Zhang, Lei Zhang, Qiuyuan Huang, Xiaodong He, Siwei Lyu, and Jianfeng Gao. Object-driven Text-to-Image Synthesis via Adversarial Training. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2

[20] Chen-Hsuan Lin, Ersin Yumer, Oliver Wang, Eli Shechtman, and Simon Lucey. ST-GAN: Spatial Transformer Generative Adversarial Networks for Image Compositing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[21] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision (ECCV)*, 2014. 2, 6

[22] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic Image Synthesis with Spatially-Adaptive Normalization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1

[23] Siyuan Qi, Yixin Zhu, Siyuan Huang, Chenfanfu Jiang, and Song-Chun Zhu. Human-centric Indoor Scene Synthesis Using Stochastic Grammar. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[24] Scott Reed, Zeynep Akata, Santosh Mohan, Samuel Tenka, Bernt Schiele, and Honglak Lee. Learning What and Where to Draw. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016. 1, 2

[25] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative Adversarial Text-to-Image Synthesis. In *International Conference on Machine Learning (ICML)*, 2016. 1, 2

[26] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning Structured Output Representation using Deep Conditional Generative Models. In *Advances in Neural Information Processing Systems (NeurIPS)*. 2015. 3

[27] Erik B Sudderth, Antonio Torralba, William T Freeman, and Alan S Willsky. Learning hierarchical models of scenes, objects, and parts. In *IEEE International Conference on Computer Vision (ICCV)*, 2005. 1

[28] Kai Wang, Manolis Savva, Angel X Chang, and Daniel Ritchie. Deep convolutional priors for indoor scene synthesis. *ACM Transactions on Graphics (TOG)*, 2018. 2

[29] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[30] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris Metaxas. StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 1, 2

[31] Bo Zhao, Lili Meng, Weidong Yin, and Leonid Sigal. Image Generation from Layout. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2, 7