Learning Graph Variational Autoencoders with Constraints and Structured Priors for Conditional Indoor 3D Scene Generation

Aditya Chattopadhyay[1]*, Xi Zhang[2], David Paul Wipf[2], Himanshu Arora[2], and René Vidal[2]
[1]Johns Hopkins University, MD, USA, [2]Amazon.com, Inc.
achatto1@jhu.edu, {xizhn, daviwipf, arorah, rvidal}@amazon.com

Abstract

We present a graph variational autoencoder with a structured prior for generating the layout of indoor 3D scenes. Given the room type (e.g., living room or library) and the room layout (e.g., room elements such as floor and walls), our architecture generates a collection of objects (e.g., furniture items such as sofa, table and chairs) that is consistent with the room type and layout. This is a challenging problem because the generated scene needs to satisfy multiple constrains, e.g., each object should lie inside the room and two objects should not occupy the same volume. To address these challenges, we propose a deep generative model that encodes these relationships as soft constraints on an attributed graph (e.g., the nodes capture attributes of room and furniture elements, such as shape, class, pose and size, and the edges capture geometric relationships such as relative orientation). The architecture consists of a graph encoder that maps the input graph to a structured latent space, and a graph decoder that generates a furniture graph, given a latent code and the room graph. The latent space is modeled with autoregressive priors, which facilitates the generation of highly structured scenes. We also propose an efficient training procedure that combines matching and constrained learning. Experiments on the 3D-FRONT dataset show that our method produces scenes that are diverse and are adapted to the room layout.

1. Introduction

The last few years have seen significant advances in image generation powered by the emergence of deep generative models such as GANs [11] and VAEs [16]. State-of-the-art methods are able to generate images of a single object category (e.g., faces) with amazingly realistic quality (e.g., [14, 36]). However, the problem of generating images of complex scenes composed of multiple objects in diverse arrangements remains a challenge. As an example, images of indoor scenes consist of room elements (floor, walls, etc.) and furniture items (table, chairs, beds, etc.) arranged in different ways depending on the room type (living room, bedroom, etc.). Moreover, room elements and furniture items should satisfy geometric constraints, e.g., each object must lie inside the room and on the floor, two objects cannot occupy the same volume, some objects tend to co-occur in particular orientations relative to the room layout.

To address some of the challenges, recent work on indoor scene image generation [10] uses GANs with multiple discriminators that specialize in localizing different objects within an image. By adding a "broker" to mediate among such discriminators, [10] achieves state-of-the-art (SOTA) results on synthesizing images of living rooms. However, such SOTA image generation models are far from capturing the rich structure present in indoor scenes. For example, [26] notice that these models fail to respect the relationships between scene objects and often cannot preserve certain shapes like axis-aligned polygons. We contend that addressing such complex image generation problems requires reasoning about the scene content in 3D space.

As a stepping stone, this paper focuses on the problem of conditional generation of the scene's 3D layout, rather than a 2D image, though we can synthesize images using a renderer given the layout. Specifically, we assume we are given the room type (e.g., living room or bedroom) and the room layout (spatial arrangement of walls, windows and doors), and our goal is to generate a collection of furniture items (e.g., sofa, coffee table and chairs) that is consistent with the room type and layout. For example, a bedroom must consist of a bed, typically placed in the center of the longest wall in the room. Moreover, we expect the generator to synthesize diverse object arrangements for the same room. This problem of conditional 3D layout generation is important in applications such as room decoration, where the goal is to produce diverse decors for a given room.

Recent work [2, 40, 15] aims to address this problem using supervision in the form of scene hierarchies or relational graphs. However, the contextual space of possible arrangements of objects in a room is simply too large to be modeled

*This work was done during Aditya's internship with Amazon.

using hand-crafted heuristics or hierarchies. This has led to recent efforts on training networks directly from data using autoregressive models based on CNNs [33] or Transformers [41, 27]. However, while these models are adept at generating indoor scenes, they lack the advantages of a learnt latent space such as that of Variational Autoencoders (VAEs).

One advantage of using learnt latent variables is that they allow for more controlled generations. For example, users can traverse the latent space to manipulate generated samples [12, 20] (see Figure 4). Such manipulations are not easy to implement in autoregressive models which lack explicit modelling of a latent space. Another advantage of using learnt latent variables is that they are often a good representation of data which can be used to bootstrap several downstream applications. In this work, we show one such application of furniture recommendation given a floorplan by retrieving the most "appropriate" furnished room from a database curated by human designers and adapting it to the new floorplan. In this case, the learnt latent representation is used for solving the retrieval problem. However, despite these advantages, we found in our experiments that existing graph-based VAE architectures are insufficient for indoor scene generation. This observation is echoed by Para *et al*. [26] in their work on 2D layout generation, where they conjecture that current VAE architectures struggle with the discrete nature of graphs and layouts.

Paper contributions. To remedy this, we propose a graph-based VAE model for the synthesis of 3D indoor scenes conditioned on the room type and layout (floorplan). We represent both the room and furniture layouts with an attributed graph. We then present a scene generative model consisting of a graph encoder that maps the input graph and the room type to a latent space, and a graph decoder that generates a furniture graph, given a latent code and the room graph. Our model considerably reduces the performance gap between VAEs and state-of-the-art autoregressive models [27] for indoor scene synthesis. Specifically, we make the following contributions:

1. *A structured autoregressive prior for graphs*:[1] This is our main contribution. Contemporary graph-VAE architectures typically encode the graph into a single latent vector and use a multi-layered perceptron (MLP) to decode it back to a graph [34, 17]. In contrast, we propose to have a separate latent code for each furniture item. While the use of an i.i.d. Gaussian prior for each latent code had been previously explored in [24], this limits performance since the graph decoder struggles to learn complex relationships between different furniture nodes from i.i.d. latent codes.[2] Instead, we propose a novel autoregressive prior based on linear Gaussian models which allow for learning a dependency structure among different latent variables.
2. *Learning graph-VAEs with autoregressive priors under constraints:* To learn our structured graph-VAE model, we propose an efficient way to compute the KL divergence term in the VAE objective which requires a matching procedure since there is no canonical ordering of graph nodes. To facilitate learning, we use simple intuitive constraints like limiting the relative distances between furniture items, such as a chair and a table. These can be easily computed from training data. We then train our graph-VAE model under these constraints utilizing a recently introduced constrained learning framework [4].
3. *Experimental evaluation*: Our quantitative and qualitative experiments show that our method outperforms state-of-the-art graph VAE architectures, bridging the performance gap between latent-variable models and autoregressive models for the task of 3D scene synthesis. Moreover, we show a unique application of our latent-variable model which is not possible with autoregressive models. Finally, we show how one can edit the generated scenes post-hoc by traversing the latent space.

2. Related Work

Graph-based inference. Graphical representation of scenes and graph-based inference have been extensively studied in the past. Early works [7, 8, 42, 13, 31] employed "shallow" methods like hand-crafted graph kernels or probabilistic graphical models to learn the furniture arrangements. Recent works leverage deep generative models to learn good scene representations directly from spatial data. The community has explored avenues for combining graphs with VAEs to synthesize 3D scenes [23, 43, 24]. However, all these methods rely on strong heuristics on defining object relations. For instance, [24] relies on user-defined scene-graphs as input, [23] requires hand-crafted hierarchies, and [29] uses heuristics to extract context-free grammars from data which are then used to train a grammar-VAE [21]. Our proposed VAE (inspired from [25]) is different from all these methods in that we do not use any such strong heuristics on object relations, but only a check of association between a furniture item and a room element using a distance measure. On the other hand, Wang *et al*. [39] uses graphs for high-level planning of the furniture layout of the room in a 2-stage approach where they train a generator to synthesize scene graphs followed by a CNN to propose consistent furniture poses. Their model has no latent variables and is slower due to the 2-stage process.

[1] Note the use of the term autoregressive here does not refer to autoregressive models which is a type of generative model which generates the scene iteratively, one furniture at a time by learning the conditional distribution of furniture given the scene rendered so far. The usage here refers to the prior distribution in a VAE which is autoregressive in the latent space.

[2] This is corroborated by our experiments in §4 where we consider this architecture as Baseline B1.

Autoregressive scene generation. Recent successful models for indoor scene synthesis are all autoregressive in nature [40, 33, 41, 27]. Wang et al. [41] introduced an autoregressive scene generation pipeline called Sceneformer, which uses multiple transformers [37] to predict the objects' category, location and size separately. Concurrently, FastSynth [33] introduced a similar pipeline, where the authors train separate CNNs based on a top-down representation of the scene to sequentially insert objects into the scene. Their method however requires auxilary supervision in the form of depth and semantic segmentation maps. Another recent transformer-based autoregressive approach was proposed in [27], which replaces the multiple trained models of past works with a single unified model. Unlike these models, we learn an end-to-end latent variable model to generate 3D indoor scenes trained from spatial data.

Expressive latent distributions for VAEs. There has been extensive work into designing expressive distributions for VAEs. For example, [19] proposes a hierarchical prior, [32] uses normalizing flows to model a more expressive posterior distribution over latents, [6] uses an autoregressive prior, and [35] advocates the use of a mixture distribution for the prior based on the posterior distribution of the encoder. However, learning expressive distributions for latent spaces which are expressed as graphs is challenging due to the absence of any canonical ordering between the different nodes of a graph. This makes computing the required KL divergence term in the ELBO notoriously difficult. In this work we propose to model the latent space as an autoregressive linear Gaussian model, which allows us to formulate the ordering as a quadratic assignment problem for which we also propose an efficient approximation.

3. Our Approach

This section describes the proposed model. First, we describe how we represent the 3D scene layout with an attributed graph. Next, we describe our architecture. The VAE encoder is a Graph Neural Network (GNN) that processes the graph and produces latent variables which are then passed to another GNN which serves as the VAE decoder. Then, we describe how we parameterize the prior on the latent space using an autoregressive model which is learnt. Finally, we describe the proposed training methodology, which includes some constraints for faster convergence. A schematic depiction of our approach is given in Figure 1a with more details in the Appendix.

3.1. Indoor scene representation as a graph

We represent an indoor scene as an attributed graph $G = (V, E, X)$. Here the nodes V denote room layout components (floor, wall, windows) and furniture items (sofa, chair, bed), the edges $E \subseteq V \times V$ denote relationships between the nodes (e.g., relative orientation of sofa to wall),

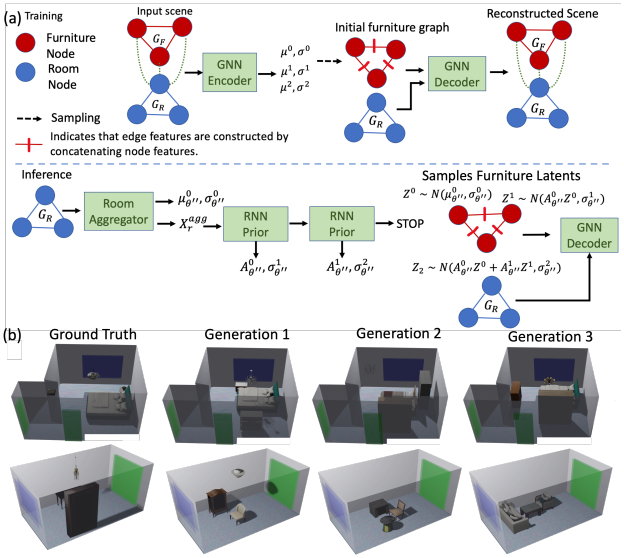

Figure 1. ***(a) Overview of the proposed model.*** *The input scene graph includes the furniture and room sub-graphs and is complete (we omit depicting certain edges to prevent clutter). The Encoder predicts a mean and variance per furniture node, which are then used to sample latents for proposing furniture nodes by the decoder. During inference we use the proposed autoregressive prior to generate the latents, which are then subsequently processed by the decoder for scene synthesis;* ***(b) Generated scenes.*** *Sample generations for our model for a bedroom (row 1) and a library (row 2). Green rectangles indicate doors while blue rectangles indicate windows*

and the attributes X denote features associated with the nodes and edges (e.g., location of furniture items or relative orientation between bed and wall). The graph has two node types (room nodes V_R, furniture nodes V_F), three edge types (room-room edges E_{RR}, room-furniture edges E_{RF} and furniture-furniture edges E_{FF}), and five attribute types corresponding to these node and edge types (X_R, X_F, X_{RR}, X_{RF} and X_{FF}). In this work, we consider the graph as complete, that is, $E_{RF} = V_R \times V_F, E_{FF} = V_F \times V_F$ and $E_{RR} = V_R \times V_R$.

We will identify two main subgraphs of G. The room layout graph $G_R = (V_R, E_R, X_R, X_{RR})$ consists of $n_R := |V_R|$ nodes (or room elements) and $e_R := |E_R|$ edges, where the node attributes $X_R \in \mathbb{R}^{n_R \times d_R}$ denote the class, location, orientation and size of the room element, and the edge attributes $X_{RR} \in \mathbb{R}^{e_R \times d_{RR}}$ encode geometric or functional relationships between two room elements (relative location, relative orientation etc.). Similarly, the furniture layout graph $G_F = (V_F, E_F, X_F, X_{FF})$ consists of $n_F := |V_F|$ nodes (or furniture items) and $e_F = |E_F|$ edges, where the node attributes $X_F \in \mathbb{R}^{n_F \times d_F}$ denote the class of the furniture item, its location, orientation, size, and its 3D shape descriptor, and its edge attributes $X_{FF} \in \mathbb{R}^{e_F \times d_{FF}}$ encode geometric or functional relation-

ships between two furniture items. We obtain the shape descriptors of each furniture item by processing their 3D point clouds through PointNet [30] pretrained on ShapeNet [5]. The room type T is given as input to the graph encoder as a categorical variable.

3.2. Proposed Generative Model

We would like to design and learn a probabilistic model $p(G_F \mid n_F, G_R, T)$ that generates a furniture layout G_F given the number of furniture items n_F, the room layout G_R and the room type T (say, bedroom or library room). We assume there exists a latent variable Z such that

$$p(G_F \mid n_F, G_R, T) = \int p(G_F \mid Z, n_F, G_R, T) p(Z \mid n_F, G_R, T) dZ. \quad (1)$$

Our proposed model consists of three main ingredients:

1. An encoder, $q_\phi(Z \mid n_F, G, T)$, which maps the number of furniture items, the room and furniture layouts (G_R and G_F resp.) and the room type to a latent variable Z, which captures the diversity of room-aware furniture layouts. The parameters of the encoder are denoted as ϕ.
2. A decoder, $p_{\theta'}(G_F \mid Z, n_F, G_R, T)$, which maps the latent variable, the number of furniture items as well as the room layout and type to a furniture layout. The parameters of the decoder are denoted as θ'.
3. A prior model $p_{\theta''}(Z \mid n_F, G_R, T)$. The difference between the prior model and the encoder is that the prior model only considers the room layout G_R and not G. The parameters of the prior model are denoted as θ''.

The encoder, decoder and prior model are parameterized with GNNs which we will describe in the next paragraphs.

Graph encoder. The encoder models the approximate posterior of a latent variable Z given (n_F, G, T). We assume that the distribution of Z, $q_\phi(Z \mid n_F, G, T)$, is Gaussian with mean $\mu_\phi(n_F, G, T)$ and a diagonal covariance matrix with diagonal entries $\sigma_\phi(n_F, G, T)$. The distribution parameters (μ_ϕ, σ_ϕ) are modeled as the output of an attention-based message passing graph neural network (MP-GNN) with weights ϕ. The design of the MP-GNN is inspired by [25], where each layer $l = 1, \ldots, L$ of the MP-GNN maps a graph G^{l-1} to another graph G^l by updating the graph's node and edge features. Specifically, let h_i^l and h_{ij}^l denote the features of node i and edge (i, j) of graph G^l, respectively. Let the input to the network be the graph $G^0 = G$, so that h_i^0 and h_{ij}^0 denote the node features (rows of X_R and X_F) and edge features (rows of X_{RR}, X_{FF} and X_{RF}), respectively. At each iteration of node and edge refinement, the MP-GNN updates both node and edge features as a function of their features and neighboring features in the previous layer using an attention mechanism that depends on the node and edge type. More details on the specific update functions used are given in Appendix §A.2.1.

After L layers of refinement, we obtain a graph G^L whose node features are mapped via a linear layer with weights W_μ, W_σ to obtain the parameters of the Gaussian model as $\mu_\phi^i(G, T) = W_\mu h_i^L$, $\sigma_\phi^i(G, T) = \exp(W_\sigma h_i^L)$ where $i = 1, \ldots, n_F$. Note that there is a different Gaussian for each node of the graph. Therefore, the output of the encoder will be two matrices $\mu_\phi(G, T, S) \in \mathbb{R}^{n_F \times d_F}$ and $\sigma_\phi(G, T, S) \in \mathbb{R}^{n_F \times d_F}$ corresponding to the mean and standard deviation vectors of the latent variable matrix $Z \in \mathbb{R}^{n_F \times d_F}$.

Graph decoder. The decoder maps (Z, n_F) and the room layout and type (G_R, T) to a desired furniture layout via the distribution $p_{\theta'}(G_F \mid Z, n_F, G_R, T)$. The generative process proceeds as follows. First, an initial fully connected furniture graph G_F^0 is instantiated. Each node of G_F^0 is associated with a feature of dimension d_F corresponding to one of the rows of Z. Each edge (i, j) of G_F^0 of type $\epsilon \in \{RF, FF\}$ is associated with a feature $Z_{ij} = (Z_i, Z_j)$ as the concatenation of the node features. As a result, we obtain an initial graph G_0 that includes both the initial furniture graph G_F^0 as well as the given room graph G_R as subgraphs. The initial graph G_0 is then passed to an MP-GNN, which follows the same operations as the encoder MP-GNN. The output of the MP-GNN is the furniture subgraph G_F^L. Each furniture node of G_F^L is then individually processed through an MLP to produce parameters for the furniture layout graph distribution $p_{\theta'}(G_F \mid Z, n_F, G_R, T)$. We assume that this distribution can be factorized as:

$$p_{\theta'}(G_F \mid Z, n_F, G_R, T) \quad (2)$$
$$= \prod_{i=1}^{n_F} p_{\theta'}(shape_i \mid Z, G_R, T) p_{\theta'}(orien_i \mid Z, G_R, T)$$
$$p_{\theta'}(loc_i \mid Z, G_R, T) p_{\theta'}(size_i \mid shape_i) p_{\theta'}(cat_i \mid shape_i),$$

where $shape_i, orien_i, cat_i$ denote features of furniture i. More specifically, we assume that given latent Z, room layout G_R and type T, the furniture items are independent of each other. For each furniture item, we further assume that shape, orientation and location features are conditionally independent. However, since the PointNet shape features implicitly capture the furniture's 3D configuration, we condition the size and category distributions on this shape feature.

We parameterize the shape and location features as a normal distribution, the size feature as a lognormal distribution (since size is always positive), and category and orientation features as categorical distributions. Since both the Encoder and Decoder graphs have n_F nodes that are in one-to-one correspondence, we can define our reconstruction loss (first term in (9)) by simply comparing their node features without the need for an explicit matching procedure.

Graph prior. Recall our latent space Z is modelled such that there is a latent variable corresponding to each furniture node in the graph. Many popular graph VAE models assume an i.i.d. normal prior for each node [18, 24]. However, such a model is restrictive for our purposes. MP-GNNs achieve permutation equivariance by sharing the weight matrices across every node in the graph. When the graph is complete, as is the case here, the marginal distribution of every output node after L GNN layers will be identical if they are initialized as i.i.d. Gaussian at the input layer. Since, the output nodes of the decoder GNN after L layers correspond to different furniture features, having identical marginals is detrimental. This claim is supported by experiments (Figure 2) where the i.i.d. prior baseline models struggle to learn proper furniture placements. To remedy this, we propose to parameterize prior distribution as an autoregressive model based on linear gaussian models [3]. More specifically,

$$p(Z^0 \mid G_R, T) = \mathcal{N}(\mu_{\theta''}(G_R, T), \sigma^0_{\theta''}(G_R, T)), \quad (3)$$
$$p(Z^i | Z^{k<i}, G_R, T) = \mathcal{N}(\sum_{k<i} A^k_{\theta''}(G_R, T) Z^k, \sigma^i_{\theta''}(G_R, T)).$$

where Z^i refers to the latent corresponding to the i^{th} furniture node. Thus, the i^{th} furniture node latent is given by a Gaussian whose mean is a linear function of all the latents $k < i$. Such a structure ensures that all the latent variables are jointly Gaussian. This allows us to analytically compute the KL divergence term and thus was favoured over more expressive probabilistic models which would introduce more stochasticity in the objective due to the need of estimating the KL divergence term via sampling. We implement (3) (see also Figure 1) with two networks:

- *Room Aggregator for* $\boldsymbol{p(Z^0 \mid G_R, T)}$*:* The room aggregator is an MP-GNN with the same architecture as the Graph Encoder, except that the input to the network is just (G_R, T) with the node and edge features initialized to X_R and X_{RF}, respectively. After L GNN layers, all the room node features are aggregated by a mean pooling operation to obtain a global representation of the room layout plan X^{agg}_R. This X^{agg}_R is then passed through an MLP to compute $\mu_{\theta''}(G_R, T)$ and $\sigma^0_{\theta''}(G_R, T)$.
- *RNN Prior for* $\boldsymbol{p(Z^i | Z^{k<i}, G_R, T)}$*:* We use a recurrent neural network to predict the matrix $A^k_{\theta''}(G_R, T)$ and the variance $\sigma^i_{\theta''}(G_R, T)$ at each node index. The RNN is initialized with X^{agg}_R. We need additional constraints on each $A^k_{\theta''}(G_R, T)$ to prevent the dynamics model in (3) from diverging to infinity. This is typically done by controlling the spectral radius or its proxy, the spectral norm [22], of the matrices $\{A^k_{\theta''}(G_R, T) : k \in [1, 2, ..., n_F]\}$. Thus, the predicted matrix is taken to be $\frac{A^k_{\theta''}(G_R,T)}{||A^k_{\theta''}(G_R,T)||_2}$, where $||A||_2$ is the spectral norm of some matrix A.

3.3. Computing the KL term via matching and constrained learning

Note that the proposed autoregressive prior could in principle be re-expressed as a more traditional i.i.d. Gaussian prior, which is then passed through additional transformation layers that are not permutationally equivariant. While these additional layers could be absorbed into the decoder, the lack of equivariance would pose key challenges during training, as there is no longer a canonical ordering of the graph nodes. On the other hand, for the proposed autoregressive prior formulation, such an ordering is only required to compute the KL divergence term in the ELBO (9), which requires evaluating $p_{\theta''}(Z \mid G_R, T, n_F)$ for any Z sampled from the posterior $q_\phi(Z \mid G, T, n_F)$. By design, the KL divergence term can be expressed analytically and, as we will soon demonstrate, a compensatory permutation can be efficiently computed. In contrast, with an alternative decoder formulation (based on the additional transformation layers), the search for an appropriate ordering is instead needed for computing the VAE reconstruction term (i.e., evaluating the decoder $p_{\theta'}(G_F \mid Z, n_F, G_R, T)$ for any Z sampled from the posterior). The computation of this ordering would typically not be analytically tractable and further approximations would be required to evaluate this reconstruction term.

Computing the KL divergence term. Let us denote the set of latent variables corresponding to n_F furniture items to be placed in the room as $Z = \{Z^1, Z^2, ..., Z^{n_F}\}$. Let π denote the ordering among these variables. Given π, the likelihood of observing Z under our proposed prior is defined as

$$p_{\theta''}(Z \,|\, n_F, G_R, T; \pi) = \prod_{i=1}^{n_F} p_{\theta''}(Z^{\pi(i)} \,|\, \{Z^{\pi(j)}\}_{j<i}, G_R, T), \quad (4)$$

However, for $Z \sim q_\phi(Z \mid n_F, G, T)$ (the approximate posterior) we do not know this ordering π. Thus, given Z, we define the optimal order π^* as

$$\operatorname*{argmin}_{\pi} KL(q_\phi(Z \,|\, n_F, G, T) || p_{\theta''}(Z | n_F, G_R, T; \pi)) \,. \quad (5)$$

Recall $q_\phi(Z \mid n_F, G, T) = \prod_{i=1}^{n_F} \mathcal{N}(Z^i; \mu^i_\phi(G), \sigma^i_\phi(G))$. Since both the prior and posterior are jointly Gaussian, computing (5) reduces to solving a Quadratic Assignment Problem (QAP). For simplicity let us denote the distributions as

$$\begin{aligned} q_\phi(Z \mid n_F, G, T) &= \mathcal{N}(\mu_0, \Sigma_0) \\ p_{\theta''}(Z \mid n_F, G_R, T; \pi) &= \mathcal{N}(\tilde{\pi}\mu_1, \tilde{\pi}\Sigma_1\tilde{\pi}^T), \end{aligned} \quad (6)$$

where $\mu_0, \mu_1 \in \mathbb{R}^{n_F d_F}$, $\Sigma_0, \Sigma_1 \in \mathbb{R}^{n_F d_F \times n_F d_F}$, $\tilde{\pi} = \pi \otimes I_{d_F \times d_F}$ and d_F is the dimension of the latent variable used for each furniture node. Note $\pi \in \mathbb{R}^{n_F \times n_F}$ and the Kronecker product comes from the fact that we are only allowed to permute blocks of μ_1 and Σ_1 of size d_F and $d_F \times d_F$, respectively. In other words, we can only permute

Figure 2. *Qualitative comparison of our method with ATISS and baselines. The encoder-decoder architecture is the same for our method and baselines. Moreover, all three VAE methods have a separate latent code for each furniture but differ in the prior distribution employed. Baseline 1 uses an i.i.d. standard Gaussian prior. Baseline 2 uses the same i.i.d. Gaussian prior but with mean and variance parameters depending on the room subgraph. Ours uses an autoregressive prior. Windows and doors and indicated by white rectangles.*

latent vectors corresponding to furniture nodes as a whole and not the intra dimensions of Z within any furniture node. Note that due to the autoregressive structure Σ_1 will not be a diagonal matrix.

We show in the Appendix A.4.1 that (5) is equivalent to

$$\min_{\pi} Tr\left(\Sigma_1^{-1}\tilde{\pi}^T\left[\Sigma_0+\mu_0\mu_0^T\right]\tilde{\pi}\right)-2Tr\left(\tilde{\pi}^T\mu_0\mu_1^T\Sigma_1^{-1}\right). \tag{7}$$

Notice that (7) is a Quadratic Assignment Problem (QAP) which is known to be NP-Hard. For this we propose to use a fast approximation algorithm, called FAQ, introduced in [38]. FAQ first relaxes the optimization problem from the set of permutation matrices to the set of doubly stochastic matrices. It then iteratively proceeds by solving linearizations of the objective (7) using the Franke-Wolfe method. These linearizations reduce the QAP to a linear assignment problem (LAP), which can be efficiently solved by the Hungarian algorithm. After Franke-Wolfe terminates, we project the doubly stochastic solution back to the set of permutations by solving another LAP. FAQ has a runtime complexity that is cubic in the number of nodes per iteration which is faster than the quartic complexity of the matching procedure used in [34]. See Appendix A.4.2 for details.

We need to compute the optimal π^* for each graph in a mini-batch, and then compute the KL divergence term in (9) analytically given this ordering. We observe no significant gains in performance in running the FAQ algorithm more than 1 step per graph, which further speeds our method.

Learning under constraints: To facilitate faster convergence and also ensure fidelity of the learned solution, we enforce certain constraints on the reconstructed room. These constraints are derived from training data and do not require external annotations. Given input furniture graph G_F to the encoder and the reconstructed graph $\tilde{G}_F$ by the decoder, we enforce the relative positions of predicted furnitures in $\tilde{G}_F$ to be "close" to the ground truth relative positions in G_F. Similarly, we apply constraints on the relative position of the predicted furniture items with the room walls, windows and doors. Finally we apply a constraint penalizing the relative orientations between different furniture items from being too "far" away from the relative orientations in G. We explain these constraints more clearly in Appendix A.5. In practice, we observe that employing constraints enable faster training. This is depicted in Figure 5 (Appendix).

3.4. Training

Having described all the ingredients of our model, we now present our learning procedure. Given a training set, consisting of indoor scenes in the form of attributed graphs $\mathcal{G}=\{G_1,G_2,...,G_n\}$, we learn the parameters $\{\theta',\theta'',\phi\}$ by optimizing the following empirical average over the training set subject to constraints:

$$\max_{\theta',\theta'',\phi}\frac{1}{n}\sum_{i=1}^{n}\mathscr{L}(G_i,\theta',\theta'',\phi)\ \text{ s.t. }\ \frac{1}{n}\sum_{i=1}^{n}\text{Constr}(G_i)\leq\epsilon, \tag{8}$$

where, for any $G\in\mathcal{G}$, $\mathscr{L}(G,\theta',\theta'',\phi)$ denotes the Evidence Lower Bound (ELBO) defined as:

$$\begin{aligned}&\mathscr{L}(G,\theta',\theta'',\phi)\\&\quad=\mathbb{E}_{Z\sim q_\phi(Z|n_F,G,T)}[\log p_{\theta'}(G_F\mid Z,n_F,G_R,T)]\\&\quad-KL(q_\phi(Z\mid n_F,G,T)\,||\,p_{\theta''}(Z\mid n_F,G_R,T)).\end{aligned} \tag{9}$$

In (8), ϵ is a user-defined hyperparameter that determines how strictly the constraints are enforced. We solve (8) using the learning under constraints framework introduced by

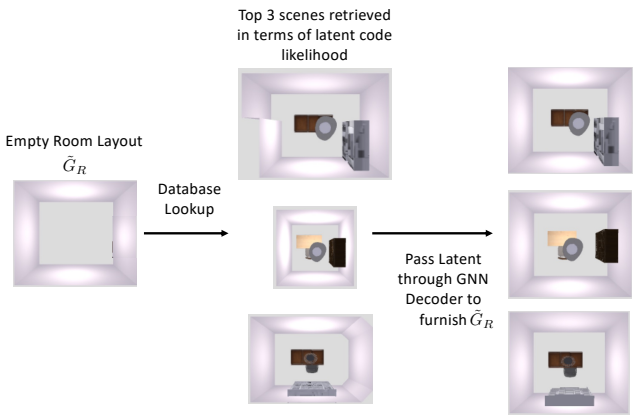

Figure 3. *Manipulating the latent space to generate room-aware designs that best match a designer-curated database.*

[4], which results in a primal-dual saddle point optimization problem. Please see Algorithm 1 (Appendix) for details.

3.5. Inference

For inference, we start with a room layout graph G_R and type T along with the number of furniture items to be placed in the room n_F. We then use the learnt autoregressive prior to sample n_F latent variables recursively. This latent Z along with (n_F, G_R, T) is processed by the graph decoder to generate the furniture layout subgraph G_F. For scene rendering, we use the predicted shape descriptor for each furniture item and perform a nearest neighbour lookup using the ℓ_2 distance against a database of 3D furniture mesh objects indexed by their respective PointNet feature. We then use the predicted size and orientation to place furniture items in the scene. Note, the inclusion of shape descriptors allows our model to reason explicitly about the furniture's 3D shape as opposed to retrieval based on just furniture category and size as in [41, 27].

4. Experiments

In this section, we evaluate the effectiveness of the proposed approach qualitatively and quantitatively. Please see Appendix A for additional figures and details.

Dataset. We use the 3D-FRONT dataset [9] for all our experiments. The dataset consists of roughly $14k$ rooms, furnished with 3D mesh furniture items. We consider 4 room types (bedroom, living room, library and dining room), which contain furniture items from 34 categories.

Training protocols. We use an 80-20 train-test split to train all models. Since our model predicts PointNet shape descriptors, which can be used to retrieve all 34 categories, we train our model to predict only 7 "super-categories"[3] of furniture items.[4] Since furniture is mostly axis-aligned, we discretize the orientation space into four categories $[0°, 90°, 180°, 270°]$. We also rotate the room by multiples of $90°$ as data augmentations. We train all models using the ADAM optimizer for 1,500 epochs with a batch size of 128.

Table 1. *Quantitative comparison of our method with other models and baselines. Real vs. Generated Scene Discrimination Accuracy closer to 0.5 is better.*

	Category KL Divergence (↓)				Real vs. Generated Scene Discrimination Accuracy			
	Bedroom	Living	Dining	Library	Bedroom	Living	Dining	Library
ATISS	**0.01**	**0.00**	**0.01**	**0.01**	**0.75**	**0.82**	**0.75**	0.86
Baseline B1	0.02	0.02	0.05	0.09	0.94	0.93	0.90	0.91
Baseline B2	0.02	0.02	0.05	0.07	0.93	0.91	0.85	0.90
Ours	**0.01**	0.01	0.02	0.02	0.83	0.88	0.78	**0.82**

Baselines. We compare our VAE model with two baselines (inspired from [24]), which have the same Encoder-Decoder architecture but employ the commonly used i.i.d. prior: (i) **Standard Prior (B1):** Each $Z^i \sim \mathcal{N}(0, I)$ for $i \in [1, ..., n_F]$, (ii) **Non-autoregressive Learnt Prior (B2):** Each $Z^i \sim \mathcal{N}(\mu(G_R), \sigma(G_R))$. The mean and variance parameters of this distribution are learnt by an MPGNN, which processes just the room subgraph.

Scene generation. In Figure 1b, we illustrate the effectiveness of the proposed method at generating diverse furniture recommendations given a room layout. Notice how our model generates diverse furniture arrangements and 3D appearances e.g. sampled beds, cabinets and ceiling lights look distinct in row 1. In the library (row 2), our model proposes diverse arrangements such as a simple study room or a lounge with a couch and chair. In Figure 2 we qualitatively compare our generations for different rooms with ATISS and VAE baselines. The figure shows that our baseline models have inconsistent overlapping furniture placements. In contrast, our model generates plausible furniture arrangements, bridging the performance gap between latent-variable models and autoregressive ones (ATISS).

In Table 1, we provide quantitative metrics comparing our model with baseline VAEs and ATISS. The first metric, Category KL Divergence, measures how well the model captures the frequency of categories in an indoor scene using all 34 fine-grained furniture labels when compared to the ground truth. We obtain the fine-grained label of a generated furniture as the category of the nearest-neighbor furniture retrieved using the predicted shape features. On this metric, our method is competitive with ATISS showing that even though we explicitly model "super-categories" only, our model is able to capture well the fine-grained object label frequencies of the test set via the shape descriptors.

The second metric, real vs generated scene discrimina-

[3] Cabinet/Shelf, Bed, Chair, Table, Sofa, Pier/Stool, Lighting.

[4] This is contrary to prior work [27] which explicitly models the 34 fine-grained categories in 3D-FRONT. We made this design choice since we use shape descriptors for each furniture item which has the fine-grained label information encoded in them. The 7 super-categories are only used to provide high-level supervision to distinguish between furnitures which may have similar shapes, for example, a chair and a sofa-chair. During synthesis, our model predicts the shape descriptors, which is used for rendering, and thus can generate furniture's belonging to all the 34 categories.

tion accuracy (close to 0.5 is better), tests the ability of a network to distinguish between real and synthetic scenes. This network is trained on a dataset comprised of real scenes from 3D-FRONT and synthesized scenes using the model to be evaluated (for example, ATISS). If the generations are perfect, this trained network would not be able to distinguish between real and synthesized scenes, and the resulting classification decision would be akin to an unbiased coin flip, which corresponds to an accuracy of 0.5. Different from prior work [27], we evaluate this metric by training a GNN directly on the synthesized 3D scene graphs vs. ground truth graphs, instead of classifying the rendered 2D top-down orthographic projections. This is because at the image level, CNNs are not able to explicitly reason about 3D spatial relationships compared to a GNN. Moreover, classifying 2D projections is sensitive to the rendering used and thus not comparable across different works. On this metric, our method performs significantly better than baselines (7-10% improvement) and is competitive with ATISS.

Manipulating the latent space to generate room-aware designs that best match a designer-curated database. In the previous sub-section we showed our model's ability to generate indoor scenes given room layouts. However, apart from AI-synthesized recommendations one might also wish to obtain recommendations from human interior designers. This can be automated by having a large database curated using hand-designed interior rooms and then given an empty floor-plan by the user, recommend "appropriate" designs from this database. However, the current practice is for the user to manually browse through configurations to find a match [28, 1]. Instead, through our learnt latent space, we can use our model to suggest the best designs from the database to choose from. To simulate this, we consider the 3D-FRONT training set as our database. We use our learnt GNN encoder to convert each of these scenes into their corresponding latent code and store in the database. Then, given an empty room layout $\tilde{G}_R$ we retrieve the latent code with the highest likelihood under our autoregressive prior (3) (conditioned on $\tilde{G}_R$ and its type).[5] Finally, we pass this latent through the GNN decoder along with $\tilde{G}_R$. This ensures that the empty room is furnished according to the design of the retrieved scene while respecting the constraints imposed by $\tilde{G}_R$, e.g. furniture items should not go outside walls. Figure 3 shows results for this experiment.

Scene editing. Our model also allows the user to traverse along the latent space to edit the scene content post-generation. Given a synthesized scene, we can convert a given furniture with label c_1 to another furniture labelled c_2 ensuring all other objects are approximately in the same geometric configuration. This is a non-trivial problem since depending on the room layout we also need to decide the size and orientation of the new furniture labelled c_2. In our formulation, this can be done by finding a latent direction $v := \frac{\mu_2 - \mu_1}{||\mu_2 - \mu_1||_2}$ that transforms furniture from label $c_1 \rightarrow c_2$, where $\{\mu_1, \mu_2\}$ are the sample mean of the latent representations of furnitures with label $\{c_1, c_2\}$. Specifically, recall the GNN encoder in our model maps the input graph to a latent space with every furniture node having a separate latent code. We first compute the latent embeddings for all scenes in the training set. We then compute μ_i by averaging over all latent codes corresponding to furnitures with label c_i across all scenes in the training set, where $i \in \{1, 2\}$. Now at inference time, we synthesize a scene S with a furniture labelled c_1 using our autoregressive prior. Post-synthesis we can select the latent $\hat{z}$ corresponding to furniture c_1 and translate it to $\hat{z}' = \hat{z} + \alpha v$, where α controls the magnitude of morphing c_1 into c_2. This updated latent $\hat{z}'$ along with the latent codes of all other furniture items in S are passed through the GNN decoder again. The end result, furniture c_1 gets morphed into c_2 while keeping the relative spatial arrangement of all other furniture's the same. This process is elucidated in Figure 4.

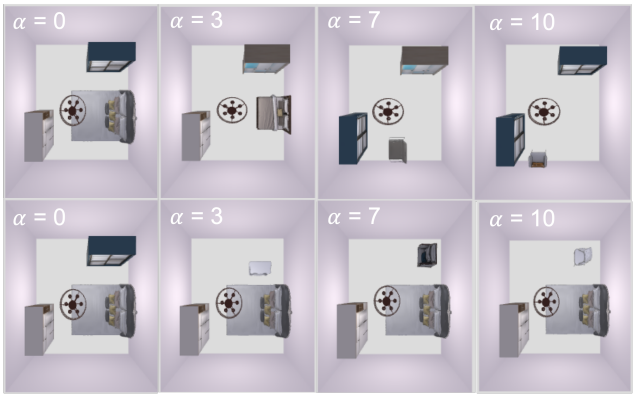

Figure 4. ***Scene editing.*** *Col 1 is a scene generated by our model. As described in the text, by changing the α parameter, in row 1 we morph the bed into a chair, and in row 2 we morph the top cabinet into a chair.*

Generation time. The average run-time for scene synthesis for our method is **130.26ms** vs. 148.51ms for ATISS (measured on NVIDIA GeForce GTX 2080 Ti machine).

5. Conclusion

We have presented a latent-variable model for generating 3D indoor scenes given the room type and layout. The proposed model outperforms existing graph VAE models and is competitive with purely autoregressive models. Moreover, we showed how a learnt latent space can be used to recommend designs from a database. In future work, we wish to explore more expressive non-linear autoregressive priors to further improve the quality and diversity of generations.

[5] We evaluate (5) to compute the ordering π of the latents for evaluating its likelihood under the autoregressive prior. More details in Appendix.

References

[1] Havenly: Online interior design and home decorating, 2022.

[2] Iro Armeni, Zhi-Yang He, JunYoung Gwak, Amir R Zamir, Martin Fischer, Jitendra Malik, and Silvio Savarese. 3D scene graph: A structure for unified semantics, 3D space, and camera. In *IEEE/CVF International Conference on Computer Vision*, pages 5664–5673, 2019.

[3] Christopher M Bishop. Pattern recognition. *Machine learning*, 128(9), 2006.

[4] Luiz Chamon and Alejandro Ribeiro. Probably approximately correct constrained learning. *Advances in Neural Information Processing Systems*, 33, 2020.

[5] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[6] Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. *arXiv preprint arXiv:1611.02731*, 2016.

[7] Matthew Fisher, Daniel Ritchie, Manolis Savva, Thomas Funkhouser, and Pat Hanrahan. Example-based synthesis of 3d object arrangements. *ACM Transactions on Graphics (TOG)*, 31(6):1–11, 2012.

[8] Matthew Fisher, Manolis Savva, and Pat Hanrahan. Characterizing structural relationships in scenes using graph kernels. In *ACM SIGGRAPH 2011 papers*, pages 1–12. 2011.

[9] Huan Fu, Bowen Cai, Lin Gao, Ling-Xiao Zhang, Jiaming Wang, Cao Li, Qixun Zeng, Chengyue Sun, Rongfei Jia, Binqiang Zhao, et al. 3d-front: 3d furnished rooms with layouts and semantics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10933–10942, 2021.

[10] Raghudeep Gadde, Qianli Feng, and Aleix M. Martinez. Detail me more: Improving gan's photo-realism of complex scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 13950–13959, October 2021.

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[12] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

[13] Chenfanfu Jiang, Siyuan Qi, Yixin Zhu, Siyuan Huang, Jenny Lin, Lap-Fai Yu, Demetri Terzopoulos, and Song-Chun Zhu. Configurable 3d scene synthesis and 2d image rendering with per-pixel ground truth using stochastic grammars. *International Journal of Computer Vision*, 126(9):920–941, 2018.

[14] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[15] Mohammad Keshavarzi, Aakash Parikh, Xiyu Zhai, Melody Mao, Luisa Caldas, and Allen Yang. Scenegen: Generative contextual scene augmentation using scene graph priors. *arXiv preprint arXiv:2009.12395*, 2020.

[16] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2014.

[17] Thomas Kipf and Max Welling. Variational graph auto-encoders. *ArXiv*, abs/1611.07308, 2016.

[18] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[19] Alexej Klushyn, Nutan Chen, Richard Kurle, Botond Cseke, and Patrick van der Smagt. Learning hierarchical priors in vaes. *arXiv preprint arXiv:1905.04982*, 2019.

[20] Abhishek Kumar, Prasanna Sattigeri, and Avinash Balakrishnan. Variational inference of disentangled latent concepts from unlabeled observations. *arXiv preprint arXiv:1711.00848*, 2017.

[21] Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. In *International Conference on Machine Learning*, pages 1945–1954. PMLR, 2017.

[22] Seth L Lacy and Dennis S Bernstein. Subspace identification with guaranteed stability using constrained optimization. *IEEE Transactions on Automatic Control*, 48(7):1259–1263, 2003.

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

[24] Andrew Luo, Zhoutong Zhang, Jiajun Wu, and Joshua B Tenenbaum. End-to-end optimization of scene layout. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3754–3763, 2020.

[25] Effrosyni Mavroudi, Benjamín Béjar Haro, and René Vidal. Representation learning on visual-symbolic graphs for video understanding. In *European Conference on Computer Vision*, pages 71–90. Springer, 2020.

[26] Wamiq Para, Paul Guerrero, Tom Kelly, Leonidas J Guibas, and Peter Wonka. Generative layout modeling using constraint graphs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6690–6700, 2021.

[27] Despoina Paschalidou, Amlan Kar, Maria Shugrina, Karsten Kreis, Andreas Geiger, and Sanja Fidler. Atiss: Autoregressive transformers for indoor scene synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[28] Sarah Perez. Amazon debuts showroom, a visual shopping experience for home furnishings. *Techcrunch*, 2019.

[29] Pulak Purkait, Christopher Zach, and Ian Reid. SG-VAE: Scene grammar variational autoencoder to generate new indoor scenes. In *European Conference on Computer Vision*, pages 155–171. Springer, 2020.

[30] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3D classification and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017.

[31] Siyuan Qi, Yixin Zhu, Siyuan Huang, Chenfanfu Jiang, and Song-Chun Zhu. Human-centric indoor scene synthesis using stochastic grammar. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5899–5908, 2018.

[32] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538. PMLR, 2015.

[33] Daniel Ritchie, Kai Wang, and Yu-An Lin. Fast and flexible indoor scene synthesis via deep convolutional generative models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6175–6183, 2019.

[34] Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *International Conference on Artificial Neural Networks*, pages 412–422. Springer, 2018.

[35] Jakub Tomczak and Max Welling. Vae with a vampprior. In *International Conference on Artificial Intelligence and Statistics*, pages 1214–1223. PMLR, 2018.

[36] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[38] Joshua T Vogelstein, John M Conroy, Vince Lyzinski, Louis J Podrazik, Steven G Kratzer, Eric T Harley, Donniell E Fishkind, R Jacob Vogelstein, and Carey E Priebe. Fast approximate quadratic programming for graph matching. *PLOS one*, 10(4):e0121002, 2015.

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

[40] Kai Wang, Manolis Savva, Angel X Chang, and Daniel Ritchie. Deep convolutional priors for indoor scene synthesis. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018.

[41] Xinpeng Wang, Chandan Yeshwanth, and Matthias Nießner. Sceneformer: Indoor scene generation with transformers. *arXiv preprint arXiv:2012.09793*, 2020.

[42] Yi-Ting Yeh, Lingfeng Yang, Matthew Watson, Noah D Goodman, and Pat Hanrahan. Synthesizing open worlds with constraints using locally annealed reversible jump mcmc. *ACM Transactions on Graphics (TOG)*, 31(4):1–11, 2012.

[43] Zaiwei Zhang, Zhenpei Yang, Chongyang Ma, Linjie Luo, Alexander Huth, Etienne Vouga, and Qixing Huang. Deep generative modeling for scene synthesis via hybrid representations. *ACM Transactions on Graphics (TOG)*, 39(2):1–21, 2020.