ABSTRACT

We introduce MeronymNet, a novel hierarchical approach for controllable, part-based generation of multi-category objects using a single unified model. We adopt a guided coarse-to-fine strategy involving semantically conditioned generation of bounding box layouts, pixel-level part layouts and ultimately, the object depictions themselves. We use Graph Convolutional Networks, Deep Recurrent Networks along with custom-designed Conditional Variational Autoencoders to enable flexible, diverse and category-aware generation of 2-D objects in a controlled manner. The performance scores for generated objects reflect MeronymNet’s superior performance compared to multiple strong baselines and ablative variants.

We also showcase MeronymNet’s suitability for controllable object generation and interactive object editing at various levels of structural and semantic granularity.

CCS CONCEPTS
- Computing methodologies → Hierarchical representations; Image representations.

KEYWORDS
deep generative model, object, part, hierarchical

1 INTRODUCTION

Many recent successes for generative approaches have been associated with generation of realistic scenery using top-down guidance from text [15] and semantic scene attributes [1, 30, 31, 33, 46]. Alongside, generative models for individual objects have also found a
degree of success [3, 5, 8, 13, 41, 43]. Despite these successes, the degree of controllability in these approaches is somewhat limited compared to the scene generation counterparts.

The bulk of object generation approaches specialize for a single category of objects with weakly aligned part configurations (e.g., faces [3, 8, 13]) and for objects with associated text description (e.g., birds [41, 43]). Although models such as BigGAN [5] go beyond a single category, conditioning the generative process is possible only at a category level. In addition, all of the existing approaches induce contextual bias by necessarily generating background along with the object. As a result, the generated object cannot be utilized as a sprite (mask) for downstream tasks [6] (e.g. compositing the object within a larger scene).

To address the shortcomings mentioned above, we introduce MeronymNet\(^1\), a novel unified part and category-aware generative model for object sprites (Fig. 2). Conditioned on a specified object category and an associated part list, a first-level generative model stochastically generates a part bounding box layout (Sec. 3.3). The category and generated part bounding boxes are used to guide a second-level model which generates semantic part maps (Sec. 3.4). Finally, the semantic part label maps are conditionally transformed into object depictions via the final level model (Sec. 3.5). Our unified approach enables diverse, part-controllable object generations across categories using a single model, without the necessity of multiple category-wise models. Additionally, MeronymNet’s design enables interactive generation and editing of objects at multiple levels of semantic granularity (Sec. 6). Please visit our project page http://meronymnet.github.io/ for source code, generated samples and additional material.

2 RELATED WORK

2-D object generative models: Within the 2-D realm, approaches are designed for a specific object type (e.g. faces [3, 8], birds [41, 43], flowers [29]) and usually involve text or pixel-level conditioning [16]. In some cases, part attributes are used to generate these specific object types [13, 37]. BigGAN [5] is a notable example for multi-category object generation. Apart from these works, objects are usually generated as intermediate component structures by scene generation approaches [15, 17, 30]. Unified, part-controllable multi-category object generative models do not exist, to the best of our knowledge.

3-D object generative models: A variety of part-based generative approaches exist for 3-D objects [22, 25–27, 35, 36]. Unlike our unified model, these approaches generally train a separate model for each object. Also, the number of object categories, maximum number of parts and variation in intra-category spatial articulation in these approaches is generally smaller compared to our setting.

Scene Generation: A common paradigm is to use top-down guidance from textual scene descriptions to guide a two stage approach [15, 17]. The first stage produces object bounding box layouts while the second stage component [33, 46] translates these layouts to scenes. In some cases, the first stage produces semantic maps [1, 8] which are translated to scenes using pixel model translations [16, 30]. We too employ the approach of intermediate structure generations but with three levels in the generative hierarchy and with object attribute conditioning.

Graph Convolutional Networks (GCNs): GCNs have emerged as a popular framework for working with graph-structured data and predominantly for discriminative tasks [34, 40, 44, 47]. In a generative setting, GCNs have been utilized for scene graph generation [12, 39]. In our work, we use GCNs which operate on part-graph object representations. These representations, in turn, are used to optimize a Variational Auto Encoder (VAE) generative model. GCN–VAE have been used for modelling citation, gene interaction networks [20, 38] and molecular design [24]. To the best of our knowledge, we are the first to introduce a conditional variant of GCN–VAE for multi-category object generation.

3 MERONYMNET

We begin with a brief overview of Variational Auto Encoder (VAE) [19] and its extension Conditional Variational Auto Encoder (CVAE) [32]. Subsequently, we provide a summary of our overall generative pipeline (Sec. 3.2). The sections following the summary describe the components of MeronymNet in detail (Sec. 3.3–3.5).

3.1 VAE and CVAE

VAE: Let \( p(d) \) represent the probability distribution of data. VAEs transform the problem of generating samples from \( p(d) \) into that of generating samples from the likelihood distribution \( p_{θ}(dz) \) where \( z \) is a low-dimensional latent surrogate of \( d \), typically modeled as a standard normal distribution, i.e. \( p(z) = \mathcal{N}(0, I) \). Obtaining a representative latent embedding for a given data sample \( d \) is possible if we have the exact posterior distribution \( p(z|d) \). Since the latter is intractable, a variational approximation \( q_θ(z|d) \), which is easier to sample from, is used. The distribution is modelled using an encoder neural network parameterized by \( φ \). Similarly, the likelihood distribution is modelled by a decoder neural network parameterized by \( θ \). To jointly optimize for \( θ \) and \( φ \), the so-called evidence lower bound (ELBO) of the data’s log distribution (\( \log p(d) \)) is sought to be maximized. The ELBO is given by \( \mathcal{L}(d; φ, θ) = \mathbb{E}_{q_{θ}(z|d)}[\log p_{θ}(dz)] - κKL(q_θ(z|d)||p(z)) \) where KL stands for KL-divergence and \( λ > 0 \) is a tradeoff hyperparameter.

\(^1\)Meronym is a linguistic term for expressing part-to-whole relationships.
**BoxGCN–VAE**

**CVAE:** Conditional VAEs are an extension of VAEs which incorporate auxiliary information as a part of the encoding and generation process. As a result, the decoder now models \( p_\theta(d|z, a) \) while the encoder models \( q_\phi(z|d, a) \). Accordingly, ELBO is modified as

\[
L(d; a; \phi, \theta) = \mathbb{E}_{q_\phi(z|d, a)}[\log p_\theta(d|z, a)] - KL(q_\phi(z|d, a)||p(z|a))
\]

In our problem setting, the conditioning is performed as a gating operation where auxiliary information either modulates a feature representation of input (encoder phase) or modulates the latent variable \( z \) (decoder phase). We modify CVAE and design an even more general variant where the conditioning used for encoder and decoder can be different. For additional details on VAE and CVAE, refer to the excellent tutorial by Doersch [9].

### 3.2 Summary of our approach

Suppose we wish to generate an object from category \( c (1 \leq c \leq M) \), with an associated maximal part list \( L_c \). The category \( c \) and a list of parts \( L_c \subseteq L_c \) is used to condition BoxGCN–VAE, the first-level generative model (Sec. 3.3, pink box in Fig. 2) which stochastically generates part-labelled bounding boxes. The generated part-labeled box layout \( B_{L_c} \) and \( c \) are used to condition LabelMap–VAE (Sec. 3.4) which generates a category-specific per-pixel part-label map \( S_{L_c} \). Finally, the label map is conditionally transformed into an RGB object depiction \( O_{L_c} \) via the final-level Label2obj model (Sec. 3.5). Next, we provide architectural details for modules that constitute MeronymNet.

### 3.3 BoxGCN–VAE

**Representing the part bounding box layout:** We design BoxGCN–VAE (Fig. 3) as a Conditional VAE which learns to stochastically generate part-labelled bounding boxes. To obtain realistic layouts, it is important to capture the semantic and structural relationships between object parts in a comprehensive manner. To meet this requirement, we model the object layout as an undirected graph. We represent this graph in terms of Feature Matrix \( X \) and Adjacency Matrix \( A \) [20, 45]. Let \( p \) be the maximum possible number of parts across all object categories, i.e. \( p = \max_c \text{length}(L_c), 1 \leq c \leq M \). \( X \) is a \( p \times 5 \) matrix where each row \( r \) corresponds to a part. A binary value \( p_r \in \{0, 1\} \) is used to record the presence or absence of the part in the first column. The next 4 columns represent bounding box coordinates. For categories with part count less than \( p \) and for absent parts, the rows are filled with 0s. The \( p \times p \) binary matrix \( A \) encodes the connectivity relationships between the object parts in terms of standard part-of (meronymic) relationships (e.g. nose is a part of face) [2]. Thus, we obtain the object part bounding box representation \( \mathcal{G} = (X, A) \).

**Graph Convolutional Network (GCN):** A GCN processes a graph \( \mathcal{G} \) as input and computes hierarchical feature representations for each graph node. The feature representation at the \( (i + 1) \)-th layer of the GCN is defined as \( H_{i+1} = f(H_i, A) \) where \( H_i \) is a \( p \times F_i \) matrix whose \( j \)-th row contains the feature representation for node indexed by \( j \) \( (1 \leq j \leq p) \). \( f \) represents the so-called propagation rule which determines the manner in which node features of the previous layer are aggregated to obtain the current layer’s feature representation. We use the following propagation rule [21]:

\[
f(H_i, A) = \sigma(D^{-\frac{1}{2}} \widetilde{A} D^{-\frac{1}{2}} H_i W_i)
\]

where \( \widetilde{A} = A + I \) represents the adjacency matrix modified to include self-loops, \( D \) is a diagonal node-degree matrix (i.e. \( D_{jj} = \sum_k \tilde{A}_{jk} \)) and \( W_i \) are the trainable weights for \( i \)-th layer. \( \sigma \) represents a non-linear activation function. Note that \( H_0 = X \) (input feature matrix).

**Encoding the graph feature representation:** The feature representation of graph \( \mathcal{G} \) obtained from GCN is then mapped to the parameters of a diagonal Gaussian distribution \( \mu_{BB}(\mathcal{G}), \sigma_{BB}(\mathcal{G}), \).

---

**Figure 3:** Architecture for BoxGCN–VAE which generates part-labelled bounding box object layouts (Sec. 3.3). The numbers within the rounded light pink rectangles indicate dimensionality of fully connected layers. Generative path is shown in dark pink.
i.e. the approximate posterior. This mapping is guided via category-level conditioning (see Fig. 3). In addition, the mapping is also conditioned using skip connection features. These skip features are obtained via 1x1 convolution along the spatial dimension of bounding box sub-matrix of input (see top part of Fig. 3). In addition to part-focused guidance for layout encoding, the skip-connection also helps to avoid imploding gradients.

**Reconstruction:** The sampled latent variable $z$, conditioned using category and part presence variables ($c, l$), is mapped by the decoder to the components of $G = (X, A)$. Denoting $X = [X_1|X_{bb}]$, the binary part-presence vector $X_1$ is modelled as a factored multivariate Bernoulli distribution

$$p_{b_0}(X_1|z, c, l) = \prod_{k=1}^P D_{b_k}^{-\hat{X}_k}(1 - D_{b_k})^{1-\hat{X}_k}$$

where $D_{b_k}$ is the corresponding output of the decoder. To encourage accurate localization of part bounding boxes $X_{bb}$, we use two per-box instance-level losses: mean squared error $L_{MSE} = \sum_{i=1}^4 (X_{bb}^i - \hat{X}_{bb}^i)^2$ and Intersection-over-Union between the predicted ($X_{bb}$) and ground-truth ($\hat{X}_{bb}$) bounding boxes $L_{IOU} = -\ln(IoU(X_{bb}^i, \hat{X}_{bb}^i))$ [42]. To impose additional structural constraints, we also use a pairwise MSE loss defined over distance between centers of bounding box pairs. Denoting the ground-truth Euclidean distance between centers of $m$-th and $n$-th bounding boxes as $d_{m,n}$, the pairwise loss is defined as $L_{MSE-c} = (d_{m,n} - \hat{d}_{m,n})^2$ where $\hat{d}_{m,n}$ refers to predicted between-center distance.

For the adjacency matrix $(A)$, we use binary cross-entropy $L_{BCE}$ as the per-element loss. The overall reconstruction loss for the BoxGCN–VAE decoder for a given object can be written as:

$$L_{box} = \frac{\sum_{p}^P \sum_{x=1}^{X} \sum_{c=1}^{C} \sum_{l=1}^{L} \sum_{z=1}^{Z} L_{BCE}(p, x, c, l, z)}{X^{P/C/L/Z}}$$

It is important to note that unlike scene graphs [12, 39], spatial relationships between nodes (parts) in our object graphs are not explicitly specified. This makes our part graph generation problem setting considerably harder compared to scene graph generation.

Also note that the decoder architecture is considerably simpler compared to encoder. As our experimental results shall demonstrate (Sec. 4), the conditioning induced by category and part-presence, combined with the connectivity encoded in the latent representation $z$, turn out to be adequate for generating the object bounding box layout despite the absence of graph unpooling layers in the decoder.

### 3.4 LabelMap–VAE

Unlike bounding boxes which represent a coarse specification of the object, generating the object label map requires spatial detail for each part to be determined at pixel level. To meet this requirement, we design LabelMap–VAE as a Conditional VAE which learns to stochastically generate per-part spatial masks (Fig. 4). To guide mask generation in a category-aware and layout-aware manner,
we use feature embeddings of object category \(c\) and bounding box layout \(B_t\) generated by BoxGCN–VAE (Sec. 3.3).

**Conditional encoding of part masks:** During encoding, the binary mask for each part is resized to fixed dimensions. The per-part CNN-encoded feature representations of individual part masks are aggregated using a bi-directional Gated Recurrent Unit (GRU) (color-coded blue in Fig. 4). The per-part hidden-state representations from the unrolled GRU are stacked to form a \(p \times h_b\) representation \(H_b\).

In parallel, feature representations for each part bounding box \(B_t\) (generated by BoxGCN–VAE) are obtained using another bi-directional GRU (color-coded purple in Fig. 4). An aggregation and stacking scheme similar to the one used for part masks is used to obtain a \(p \times h_b\) feature matrix. Using a \(1 \times 1\) convolution, the latter is transformed to a \(p \times h_b\) representation \(H_b\) (see Fig. 4). To induce bounding-box based conditioning, we use \(H_b\) to multiplicatively gate the intermediate label-map feature representation \(H_c\). The result is pooled across rows and further gated using category information. In turn, the obtained feature representation is ultimately mapped to the parameter representations of a diagonal Gaussian distribution \((\mu_M, \sigma_M)\).

**Generating the part label map:** The decoder maps the sampled latent variable \(z\) to a conditional data distribution \(p_{\theta_M}(M|z, a_{bb}, a_c)\) over the sequence of label maps \(M = \{m_1, m_2, \ldots, m_p\}\) with \(\theta_m\) representing parameters of the decoder network. \(a_c\) and \(a_{bb}\) respectively represent the conditioning induced by feature representations of object category \(c\) and the stochastically generated bounding box representation \(B_t\). The gated latent vector is mapped to \(z_g\) which is replicated \(p\) times and fed to the decoder bi-directional GRU (color-coded orange in Fig. 4). The hidden state of each unrolled GRU unit is subsequently decoded into individual part masks. We model the distribution of part masks as a factored product of conditionals: \(p_{\theta_m}(M|z, a_{bb}, a_c) = \prod_{k=1}^{p} p(m_k|m_{1:k-1}, z, a_{bb}, a_c)\). In turn, we model each part mask as \(p(m_k|m_{1:k-1}, z, a_{bb}, a_c) = \text{Softmax}(Z_k^p)\) where \(Z_k^p\) represents the binary logits from \(k\)-th part’s GRU-CNN decoder.

To obtain a one-hot representation of the object label map, each part mask is positioned within a \(H \times W \times p\) tensor where \(H \times W\) represents the 2-D spatial dimensions of the object canvas. The part’s index \(k, 1 \leq k \leq p\), is used to determine one-hot label encoding while the spatial geometry is obtained by scaling the mask according to the part’s bounding box.

### 3.5 Label2obj

For translating part label maps generated by LabelMap–VAE (Sec. 3.4) to the final object depiction in a category-aware manner, we design Label2obj as a conditioned variant of the approach by Park et. al. [30] (see Fig. 5). The one-hot representation of object category is transformed via an embedding layer. The embedding output is reshaped, scaled and concatenated depth-wise with appropriately warped label maps. The resultant features comprise one of the inputs to the SPADE blocks within the pipeline. The modified loss functions for Label2obj are:

\[
L_D = -E_{l,c}[\min(0, 1 - D_G(l_c, G_C(l_c, c)))]
- E_{k,c}[\min(0, 1 + D_G(l_c, c, 0))]
L_G = -E_{l,c}[D_G(l_c, G_C(l_c, c, 0))] + \lambda_G L_{D}^{feat} + \lambda_D L_{D}^{feat}
\]

where \(L_G, L_D\) are the generator and discriminator losses, \(L_{D}^{feat}\) and \(L_{D}^{feat}\), the generator and discriminator feature matching losses [30]. The subscripts for expectation refer to RGB images \(a_c\), corresponding semantic label map \(l_c\) and object category embeddings \(c\). Our modification incorporates a category-aware discriminator [28] \((D_C)\) which complements our category-conditioned generator \((G_C)\).

### 3.6 Implementation Details

The VAE-based components (BoxGCN–VAE, LabelMap–VAE) are trained using the standard approach of maximizing the ELBO (Sec. 3.1) and with Adam optimizer [18]. For the VAE hyperparameter \(\lambda\) which trades off reconstruction loss and KL regularization term, we employ a cyclic annealing schedule [11]. This is done to mitigate the possibility of the KL term vanishing and to make use of informative latent representations from previous cycles as warm restarts. In addition, we impose a constraint over the difference of training and validation losses. Whenever the loss difference increases above a threshold (0.1 in our case), \(\lambda\) (KL regularization term) is frozen and prevented from increasing according to the default annealing schedule. BoxGCN–VAE is trained with a learning rate of \(10^{-4}\) for 300 epochs using a mini-batch of size 32. LabelMap–VAE is trained with a learning rate of \(10^{-3}\) for 110 epochs using a mini-batch size of 8. The bounding boxes generated by BoxGCN–VAE are scaled to \(500 \times 500\) while the label masks generated by LabelMap–VAE are positioned within a \(128 \times 128\) canvas.
We train Label2obj (Sec. 3.5) with 7 SPADE blocks, a mini-batch size of 16, using Adam optimizer and \(\lambda_p = \lambda_g = 10\). The initial learning rate is 2 × 10\(^{-4}\) for 8 epochs. The rate is then set to decay linearly for next 4 epochs. The output RGB image has spatial dimensions of 128 × 128.

### 4 EXPERIMENTAL SETUP

**Dataset:** For our experiments, we use the PASCAL-Part dataset [7], containing 10,103 images across 20 object categories annotated with part labels at pixel level. We select the following 10 object categories: cow, bird, person, horse, sheep, cat, dog, airplane, bicycle, motorbike. The collection is characterized by a large diversity in appearance, viewpoints and associated part counts (see Fig. 6). The individual objects are cropped from dataset images and centered. To augment images and associated part masks, we apply translation, anisotropic scaling for each part independently and also collectively at object level. We also augment via horizontal mirroring. The objects are then normalized with respect to the minimum and maximum width across all images such that all objects are centered in a [0, 1] × [0, 1] canvas. We use 75% of the images for training, 15% for validation and the remaining 10% for evaluation.

**Baseline Generative Models:** Since no baseline models exist currently for direct comparison, we designed and implemented the baselines ourselves – see Fig. 7 for a visual illustration of baselines and component configurations. We modified existing scene layout generation approach [17] to generate part layouts. In some cases, we modified existing scene generation approaches having layouts as the starting point [1, 23, 33] to generate objects. We also included modified variants of two existing part-based object generative models (3-D objects [35], faces [8]). To evaluate individual components from MeronymNet, we also designed hybrid baselines with MeronymNet components (BoxGCN—VAE, LabelMapVAE, Label2obj) included. We incorporated object category, part-list based conditioning in each baseline to ensure fair and consistent comparison. Additional details regarding the baselines can be found in project page.

**Evaluation Protocol:** For each model (including MeronymNet), we generate 100 objects for each category using the per-category part lists in the test set. We use Frechet Inception Distance [14] (FID) as a quantitative measure of generation quality. The smaller the FID, the better the quality. For each model, we report FID for each category’s generations separately and overall average FID.

### 5 RESULTS

**Qualitative results:** MeronymNet’s generations obtained from object category and associated part lists from test set can be viewed in Fig. 8. The results demonstrate MeronymNet’s ability in generating diverse object maps for multiple object categories. The results also convey our model’s ability to accommodate a large range of parts within and across object categories.

**Quantitative results:** As Table 1 shows, Meronymnet outperforms all other baselines. The quality of part box layouts from BoxGCN—VAE is better than those produced using LSTM-based LayoutVAE [17] (compare rows 1, 7, rows 3, 5 and 4, 6, also see Fig. 7). Note that the modified version of PQNet [35] (row 10) also relies on a sequential (GRU) model. The benefit of modelling object layout distributions via the more natural graph-based representations (Sec. 3.3) is evident.
The results also highlight the importance of our three-stage generative pipeline. In particular, MeronymNet distinctly outperforms approaches which generate objects directly from bounding box layouts [33, 46] (compare rows 1, 3, 4). Also, the relatively higher quality of MeronymNet’s part label maps ensures better performance compared to other label map translation-based approaches [1, 8, 16] (compare rows 1, 2, 8, 9). In particular, note that our modified variants of some models [1, 8] (rows 8, 9) employ a SPADE-based approach [30] for the final label-to-image stage.

Model scores for another generative model quality measure - Activation Maximization Score [4] - can be viewed in project page. The baseline rankings using AM Score also show trends similar to FID. The quality of sample generations from MeronymNet (second row in Fig. 9) is comparable to unseen PASCAL-Parts test images with same part list (top row). The same figure also shows sample generations from c-SBGAN, the next best performing baseline (last row). The visual quality of MeronymNet’s generations is comparatively better, especially for categories containing many parts (e.g.
person, cow, horse). Across the dataset, objects tend to have a large range in their part-level morphology (area, aspect ratio, count — Fig. 6) which poses a challenge to image-level label map generation approaches, including c-SBGAN. In contrast, our design choices — generating all part label masks at same resolution (Fig. 4), decoupling bounding box and label map geometry — all help address the challenges better [10].

It is somewhat tempting to assume that parts are to objects what objects are to scenes, compositionally speaking. However, the analogy does not hold well in the generative setting. This is evident from our results with various scene generation models as baseline components (Table 1). The structural and photometric constraints for objects are stricter compared to scenes. In MeronymNet, these constraints are addressed by incorporating compositional structure and semantic guidance in a considered, coarse-to-fine manner which enables better generations and performance relative to baselines.

Ablations: We also compute the FID score for ablative variants of MeronymNet. The results in Table 2 illustrate the importance of key design and optimization choices within our generative approach. In particular, note that the VAE version of Label2obj performs worse than our default (translation) model (Sec. 3.5). Unlike scenes, object appearances do not depict as much diversity for a specified label map. This likely makes the VAE a more complex alternative during optimization.

6 INTERACTIVE MODIFICATION

We now describe experiments used to obtain the results from the MeroBot app dialog mockup (Fig. 1). For the first scenario (panel A), we generate an object based on the user-specified category (‘cow’). If parts are not mentioned, a part list sampled from the category’s training set distribution is chosen. If the user chooses to add new parts, the GUI enables canvas drawing editor controls. The user drawn part masks and corresponding labels are merged into the part label map generated earlier. When user clicks ‘Render’ button, the updated label map is processed by Label2obj (Sec. 3.5) to obtain the updated object generation. A similar process flow, but involving specification of bounding box based edits and BoxGCN-VAE (Sec. 3.3) is used to generate a caricature-like exaggeration of the object depiction (panel B). The third panel C depicts a process flow wherein the desired edits are specified at a high-level, i.e. in terms of part presence or absence. The updated part list is subsequently used for generating the corresponding object representations (last frame). Note that the viewpoint for rendering the object has changed from the initial generation to accommodate the updated part list. This scenario demonstrates MeronymNet’s holistic, part-based awareness of rendering viewpoints best suited for various part sets.

7 CONCLUSION

In this paper, we have presented MeronymNet, a hierarchical and controllable generative framework for general collections of 2-D objects. Our model generates diverse looking RGB object sprites in a part-aware manner across multiple categories using a single unified architecture. The strict and implicit constraints between object parts, variety in layouts and extreme part articulations, generally make multi-category object generation a very challenging problem. Through our design choices involving GCNs, CVAEs, RNNs and guidance using object attribute conditioning, we have shown that these issues can be successfully tackled using a single unified model. Our evaluation establishes the quantitative and qualitative superiority of MeronymNet, overall and at individual component level. We have also shown MeronymNet’s suitability for controllable object generation and interactive editing at various levels of structural and semantic granularity. The advantages of our hierarchical setup include efficient processing and scaling with inclusion of additional object categories in future.

More broadly, our model paves the way for enabling compact, hierarchically configurable and truly end-to-end scene generation approaches. In this paradigm, the generation process can be controlled at class and part level to generate objects as we do and subsequently condition on these objects at the overall scene level. Going forward, we intend to explore modifications towards improved quality, diversity and degree of controllability. We also intend to explore the feasibility of our unified model for multi-category 3-D object generation.
