Supplementary Material: Scene Grammar Variational Autoencoder

Pulak Purkait[1[0000−0003−0684−1209]], Christopher Zach[2[0000−0003−2840−6187]], and Ian Reid[1[0000−0001−7790−6423]]

[1] Australian Institute of Machine Learning and School of Computer Science, The University of Adelaide, Adelaide SA 5005, Australia
[2] Chalmers University of Technology, Goteborg 41296, Sweden

Table of Contents

List of Figures

1 Selection of the CFG—Algorithm details

We aim to find suitable non-terminals and associated production rules that cover the entire dataset. Note that finding such a set is a combinatorial hard problem. Therefore, we devise a greedy algorithm to select non-terminals and find approximate best coverage. Let X_j, an object category, be a potential non-terminal symbol and $\mathcal{R}_j$ be the set of production rules derived from X_j in the causal graph. Let C_j be the set of terminals that $\mathcal{R}_j$ covers (essentially nodes that X_j leads to in the causal graph $\mathcal{G}$). Our greedy algorithm begins with an empty set $\mathcal{R} = \emptyset$ and chooses the node X_j and associated production rule set $\mathcal{R}_j$ to add that maximize the *gain* in coverage

$$\mathcal{G}_{gain}(\mathcal{R}_j, \mathcal{R}) = \frac{1}{|\mathcal{R}_j|} \sum_{I_i \in \mathcal{I} \setminus \mathcal{C}} |Y_i|/|I_i| \tag{1}$$

where $\mathcal{C}$ is the set of scenes that are already covered with a predefined fraction p by the set of rules $\mathcal{R}$, *i.e.*$\mathcal{C} = \{I_i \mid \frac{|Y_i|}{|I_i|} > p\}$ where Y_i is the set of terminal symbols occurs while parsing a scene I_i by $\mathcal{R}$. The algorithm continues till no further nodes and associated rule set contribute a positive gain or until the current rule set covers the entire dataset with probability p (chosen as 0.8). We name our algorithm as *p-cover* and is furnished in algorithm 2. Note that only object co-occurrences were utilized and object appearances were not incorporated in the proposed *p-cover* algorithm.

To ensure the production rules to form a CFG, we select a few vertices (anchor nodes) of the causal graph and associate a number of non-terminals. A valid production rule is *"an object category corresponding to an anchor node generates another object category it is adjacent to in $\mathcal{G}$"*. Let us consider the set of anchor nodes forms our set of non-terminals Σ. The set of all possible objects including dummy `None` is defined as the set of terminal symbols $\mathcal{V}$.

Let $\mathcal{R}_j$ be the set of production rules derived from a non-terminal X_j`inc` corresponding to an anchor object X_j, and C_j be the set of terminals that $\mathcal{R}_j$ covers (essentially nodes that X_j leads to in $\mathcal{G}$). Note that $\mathcal{R}_j$ contains mainly four types of production rules as described in [(R1)-(R4)] in the main text. A few examples of such set of production rules are displayed in Figure 5. The above strategy could lead to a large amount of production rules (ideally sum of number of the anchor points and the number of edges $|\mathcal{E}|$ in the causal graph). Note that a large number of production rules increases the problem complexity. Contrarily, an arbitrary selection of few rules leads to a small number of derivable scenes (language of constituent grammar). We propose an algorithm to find a compressed (minimal size) set of rules to cover the entire dataset. Our underlying assumption is that the distribution of objects in the test scenes is very similar to the distribution of the same in the training scenes. Hence, we use the coverage of the training scenes as a proxy for the coverage of testing scenes and derive a probabilistic covering algorithm on the occurrences of objects in the dataset.

An example snippet for the grammar produced using this algorithm on the SUNRGBD dataset [9] is as follows:

```
S → scene SCENE
SCENE → bed BED SCENE
BED → bed BED
BED → lamp BED
Bed → sofa SOFA BED
BED → pillow PILLOW BED
BED → night_stand BED
BED → dresser BED
...
BED → None
```

```
SCENE → sofa sofainc SCENE
SOFA → sofa SOFA
SOFA → pillow PILLOW SOFA
SOFA → TOWEL SOFA
SOFA → None
...
SCENE → None
```

The entire grammar is displayed in Section 5. In the above snippet the non-terminal `BED` generates another non-terminal `SOFA` that leads to an additional set of production rules corresponding to `SOFA`. Note that the grammar is right-recursive and not a regular grammar as some of the rules contain two non-terminals in the right hand side.

Note that in this grammar non-terminal symbol `BED` generates another non-terminal `SOFA` which further leads to another set of production rules corresponding to `SOFA`. The grammar is right-recursive and not a regular grammar as some of the rules contain two non-terminals in the right hand side.

2 Visualization of the latent space

To check the continuity of the latent space, the latent vectors (*i.e.*, the mean $\boldsymbol{\mu}$ of the distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$) is projected to 2D plane using data visualization algorithm t-SNE [5]. In Figure 1, we display 15 different scenes in the latent space after t-SNE projection. We observe top left and top right regions are kitchen and bathroom scenes respectively. Whereas, top and bottom regions correspond to bedroom and dining room scenes respectively. The middle region mostly corresponds to living room and office scenes. Note that proposed SG-VAE not only considers the object co-occurrences, also considers object attributes (3D pose and shape parameters). For example, two scenes consists of a *chair* and a *table* are mapped to two nearby but distinct points.

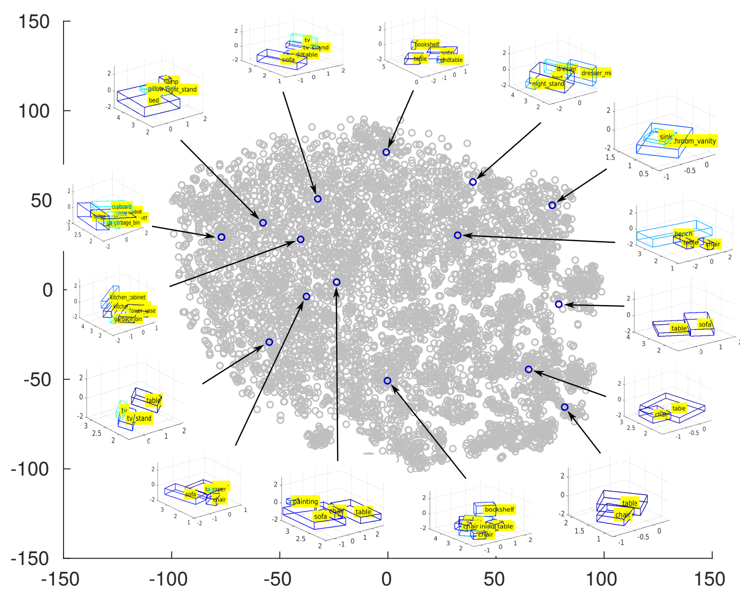


Fig. 1. We plot the 2D projection of the mean $\boldsymbol{\mu}$ of the encoded distributions of the data (encoded with 50 dimension) projected using t-SNE algorithm. 15 chosen scenes are also displayed. Note that points with similar semantic concepts are clustered around a certain region.

3 Results on SUNCG

As mentioned in the paper, due to the ongoing dispute with the SUNCG dataset, we could not include these results in the main paper. The experiment is conducted only on our local copy of the dataset for the reviewing purposes. We extracted $32,765$ bedrooms, 14814 kitchen, and $8,446$ office rooms from the dataset of $45,622$ synthetic houses. The dataset is divided into 80% training, 10% validation and remaining 10% for testing. The bounding boxes and the relevant parameters are extracted using the scripts provided by Grains [3].

Baseline comparison For visual comparison, some examples of the scene synthesized by the proposed method along with the baselines on SUNCG are shown in Figure 2.[4] Our results are similar to Grains and better than the other baselines in terms of co-occurrences and appearances (pose and shape) of different objects. A detailed 1-1 comparison with Grains is also shown in Figure 3. The quantitative comparisons are provided in the main manuscript.

Interpolation in the latent space Similar to the interpolation results on SUN RGB-D, shown in the main manuscript, an additional experiment is conducted on SUNCG dataset. Here, synthetic scenes are decoded from linear interpolations $\alpha\boldsymbol{\mu_1} + (1-\alpha)\boldsymbol{\mu_2}$ of the means $\boldsymbol{\mu_1}$ and $\boldsymbol{\mu_2}$ of the latent distributions of two separate scenes. The generated scenes are valid in terms of the co-occurrences of the object categories and their shapes and poses.

4 Additional experiments on SUN RGBD

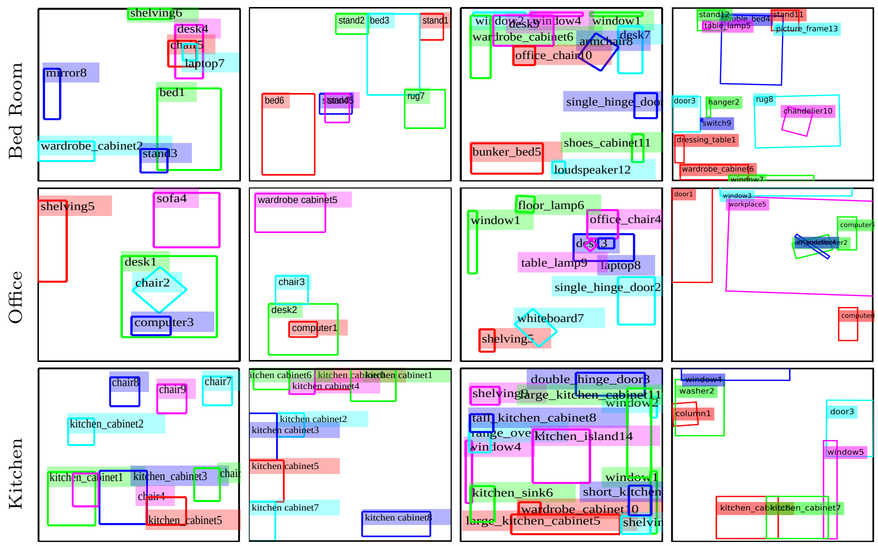
4.1 Scene layout from the RGB-D image—in detail

The task is to predict the 3D scene layout given an RGB-D image. We (linearly) map deep features (extracted from images by a DNN [13]) to the latent space of the scene-grammar autoencoder. The decoder subsequently generates a 3D scene configuration with associated bounding boxes and object labels from the projected latent vector. Since during the deep feature extraction and the linear projection, the spatial information of the bounding boxes are lost, the predicted scene layout is then combined with a bounding box detection to produce the final output.

Training Let $\mathcal{F}_i$ be the (in our case 8192-dimensional) deep feature vector extracted from the image I_i, and let $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma_i})$ be the (e.g. 50-dimensional) latent representation obtained by encoding the corresponding parse tree. We align the feature vector $\mathcal{F}_i$ with the latent distribution using a linear mapping

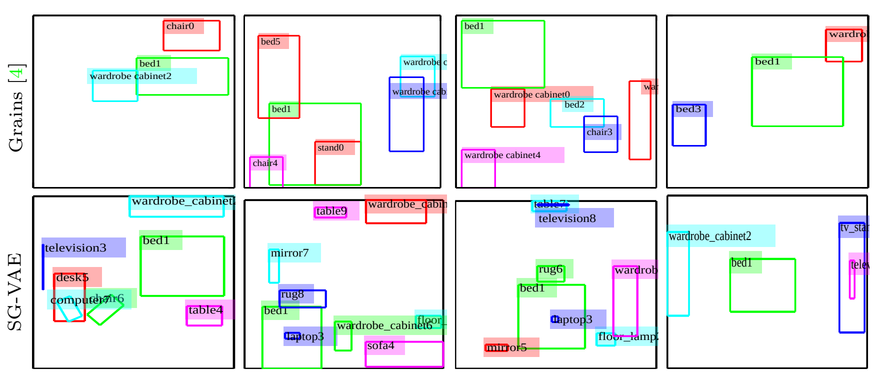
[3] https://github.com/ManyiLi12345/GRAINS

[4] We thank the authors of GRAINS [4] and HC [7] for sharing the code and the authors of FS [8] for the results displayed in Figure 2.



(a) SG-VAE (c) Grains [4] (d) FS [8] (e) HC [7]

Fig. 2. Top-views of the synthesized scenes generated by the proposed and the baseline indoor scene synthesis methods on SUNCG datasets. Note that these are just some random samples taken from the generated scenes.



Sample 1 Sample 2 Sample 3 Sample 4

Fig. 3. 1-1 comparison with Grains [4] on SUNCG dataset. Similar samples are chosen for a precise comparison. Most similar synthetic scenes are displayed. We observe that the proposed method SG-VAE produces more variety of objects in a scene than the baseline Grains.

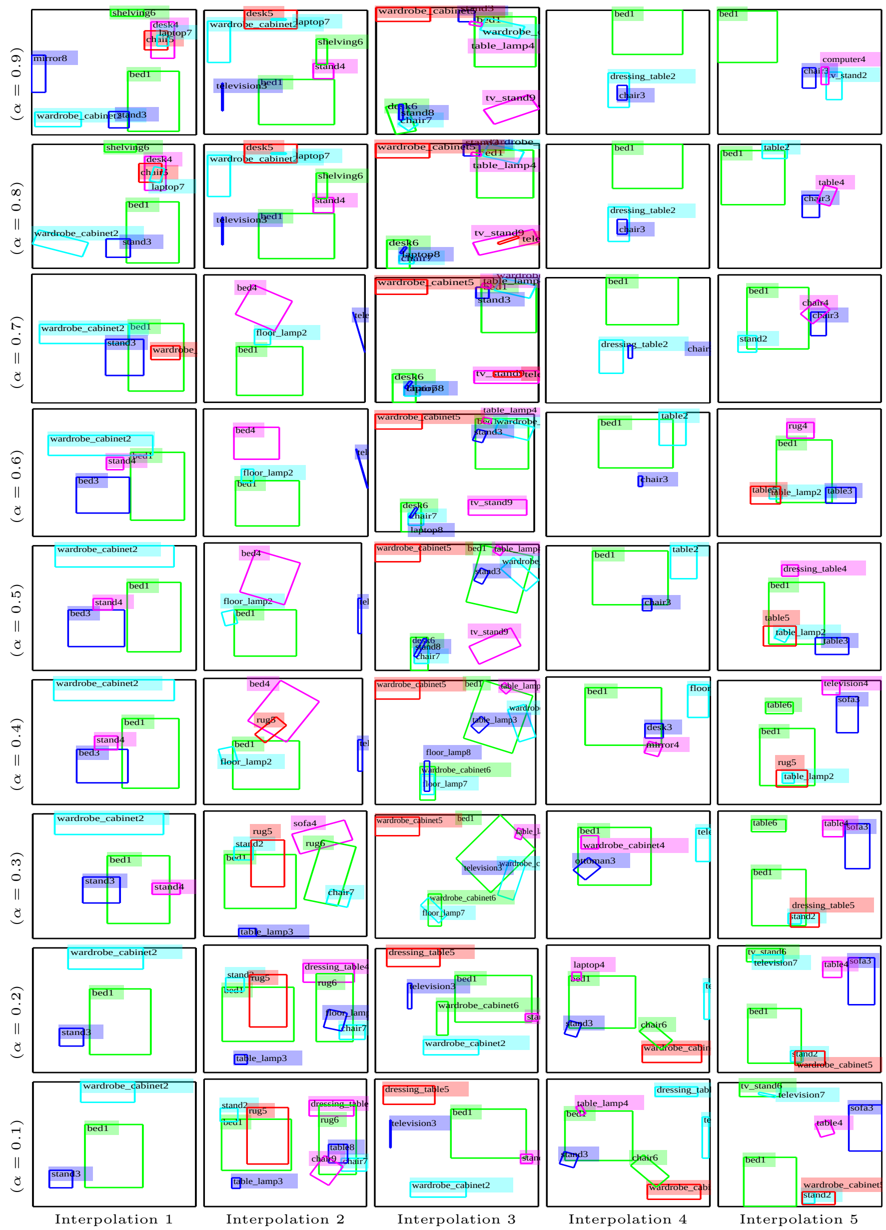


Fig. 4. Latent code interpolation on SUNCG: Synthetic scenes decoded from linear interpolations $\alpha\boldsymbol{\mu_1} + (1-\alpha)\boldsymbol{\mu_2}$ of the means $\boldsymbol{\mu_1}$ and $\boldsymbol{\mu_2}$ of the latent distributions of two separate scenes. The generated scenes are valid in terms of the co-occurrences of the object categories and their shapes and poses. The room-size and the camera view-point are fixed for better visualization. Best viewed electronically.

$\psi(\mathcal{F}_i) = A\mathcal{F}_i$, where A is a matrix to be learned from a training set $\mathcal{T} := \{I_i := (\mathcal{F}_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma_i})\}$. We minimize the cross-entropy between the predicted (deterministic) latent representation $\psi(\mathcal{F}_i)$ and the target distribution $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma_i})$, therefore the optimal matrix A is determined as $\hat{A} = \arg\min_A \sum_{I_i \in \mathcal{T}} \left(A\mathcal{F}_i - \boldsymbol{\mu}_i\right)^T \boldsymbol{\Sigma_i}^{-1}\left(A\mathcal{F}_i - \boldsymbol{\mu}_i\right) + \lambda\|A\|_2^2$. The features $\mathcal{F}_i$ of dimension 8192 are then projected into the mean of the encoded vector $\boldsymbol{\mu}_i$ (typically dimension 50). Let $\phi : \mathcal{F}_i \rightarrow \boldsymbol{\mu}_i$ be the mapping that project the feature vectors $\mathcal{F}_i$ to the latent space $\boldsymbol{\mu}_i$. A neural network could be used to learn the mapping ϕ, however, a simple linear projection is employed here, *i.e.*$\psi(\mathcal{F}_i) = A\mathcal{F}_i$. The mapping ϕ is learned from the training examples $Tr = \{I_i := (\mathcal{F}_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma_i})\}$ as follows:

$$\hat{A} = \arg\min_A \sum_{I_i \in \mathcal{T}} \Big(A\mathcal{F}_i - \boldsymbol{\mu}_i\Big)^T \boldsymbol{\Sigma_i}^{-1} \Big(A\mathcal{F}_i - \boldsymbol{\mu}_i\Big) + \lambda\|A\|_2^2 \tag{2}$$

where we also added a regularization term with weight λ (chosen as $\lambda = 100$). Differentiating the objective to zero, we get $\sum_{I_i \in \mathcal{T}} \boldsymbol{\Sigma_i}^{-1} \hat{A}(\mathcal{F}_i \mathcal{F}_i^T) + 2\lambda\hat{A} = \sum_{I_i \in \mathcal{T}} \boldsymbol{\Sigma_i}^{-1} \boldsymbol{\mu}_i \mathcal{F}_i^T$. Therefore,

$$\hat{A} = \Big(\sum_{I_i \in Tr} \mathcal{F}_i^T \boldsymbol{\Sigma_i}^{-1} \mathcal{F}_i + 2\lambda I\Big)^{-1} \sum_{I_i \in Tr} \mathcal{F}_i \boldsymbol{\Sigma_i}^{-1} \boldsymbol{\mu}_i \tag{3}$$

Note that the covariance matrix $\boldsymbol{\Sigma_i}$ is chosen to be diagonal, and thus $\hat{A}$ can be solved efficiently. The above is a system of linear equations solved by vectorizing the matrix $\hat{A}$.

Testing For test data image features [13] are extracted and then mapped to the latent space using the trained mapping $\hat{\phi}(\mathcal{F}_i) := \mathcal{F}_i\hat{A}$. The scenes are then decoded from the latent vectors $\hat{\boldsymbol{\mu}} := \mathcal{F}_i\hat{A}$ using the decoder part of the SG-VAE. The bounding box detector of DSS [10] is employed and the scores of the detection are updated based on our reconstruction as follows: the score (confidence of the prediction) of a detected bounding box is doubled if a similar bounding box (in terms of shape and pose) of the same category is reconstructed by our method. A 3D non-maximum suppression is applied to the modified scores to get the final scene layout.

4.2 Quality assessment of the autoencoder

The scenes and the bounding boxes of the test examples are first encoded to the latent representations $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma_i})$. The mean of the distributions $\boldsymbol{\mu}_i$ are then decoded to the scene with object bounding boxes and labels. The results are displayed below. Ideally, the decoder should produce a scene which is very similar to input test scene. IoU (computed over the occupied space) of the decoded scene and the original input scene is also shown in the main paper. Baseline methods for evaluation as follows:

(**BL**1) *Variant of SG-VAE*: In contrast to the proposed SG-VAE where attributes of each rule are directly concatenated with 1-hot encoding of the rule, in this variant separate attributes for each rule type are predicted by the decoder and rest are filled with zeros. *i.e.*, the 1-hot encoding of the production rules is same as SG-VAE but the attributes are represented by a $|\mathcal{R}| * \theta$ dimensional vector where $|\mathcal{R}|$ is the number of production rules and θ is the size of the attributes. For example, the pose and shape attributes $\Theta^{j \to k} = (\mathcal{P}_i^{j \to k}, \mathcal{S}_i^k)$, associated with a production rule (say pth rule) in which a non-terminal X_j yields a terminal X_k, are placed in $(p-1) * \theta + 1 : p * \theta$ dimensions of the attribute vector and rest of the positions are kept as zeros. Note that in case of SG-VAE the attributes are represented by a θ dimensional vector only.

(**BL**2) *No Grammar VAE* [2]: No grammar is considered in this baseline. The 1-hot encodings correspond to the object type is concatenated with the absolute pose of the objects (in contrast to rule-type and relative pose in SG-VAE) respectively. *i.e.*each object is represented by $|\mathcal{V}|$ dimensional 1-hot vector and a θ dimensional attribute vector. The objects are ordered in the same way as SG-VAE to avoid ambiguity in the representation. The same network as SG-VAE is incorporated except no grammar is employed (*i.e.*no masking) while decoding a latent vector to a scene layout.

(**BL**3) *Grammar VAE* [3] + *Make home* [12]: The Grammar VAE is incorporated with our extracted grammar to sample a set of coherent objects and [12] is used to arrange them. Sampled 10 times and solution corresponding to best IoU w.r.t. groundtruth is employed. Here no pose and shape attributes are incorporated while training the autoencoder. The attributes are estimated by *Make home* [12] and the best (in terms of IoU) is chosen comparing the ground-truth.

The detailed quantitative numbers are presented in the main manuscript.

Algorithm 1: Structure learning: Inductive Causation (IC) [6]

Input: a dataset $\mathcal{D}$ of natural scenes formed by a set of objects $\mathcal{V} = \{X_1, \ldots, X_n\}$

Output: a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ representing the causal relationships between the variables

Initialize $\mathcal{G} := \{\mathcal{V};\ \mathcal{E} = \mathcal{E}_0\}$; `/* Initialize with prior edges */`

for *every pair of objects* $(X_j \in \mathcal{V};\ X_{j'} \in \mathcal{V})$ **do**

  for *every conditioning variable* $X_k \in \mathcal{V} \setminus \{X_i, X_{j'}\}$ **do**

    hypothesis test $X_j \perp\!\!\!\perp X_{j'} \mid X_k$ in $\mathcal{D}$; `/* Using Conditional Algo 1 */`

  if *no independence was found* **then**

    add an undirected edge (X_j, X_j') in $\mathcal{E}$, *i.e.*, $\mathcal{E} = \mathcal{E} \cup (X_j, X_j')$.

for *every pair of objects* $(X_j \in \mathcal{V};\ X_{j'} \in \mathcal{V})$ *with a common neighbor* X_k **do**

  if $(X_j, X_j') \notin \mathcal{E}$ **then**

    if *one of* (X_k, X_j) *and* (X_k, X_j') *is directed and the other is undirected* **or**

    both are undirected **then**

      turn the triplet into a common parent structure, *i.e.*, $X_j \leftarrow\!\circ X_k \circ\!\rightarrow X_{j'}$

Propagate the arrow orientation for all undirected edges (modify the set $\mathcal{E}$ accordingly) without introducing a directed cycle. ; `/* Following Dor and Tarsi [1] */`

return $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$

Algorithm 2: p-Cover : A greedy algorithm for p-cover

Input: a dataset $\mathcal{D}$ of indoor scenes $\mathcal{I}$ formed by a set of objects $\mathcal{V} = \{X_1, \ldots, X_n\}$, the causal graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ obtained by Algorithm 1 and a probability p (chosen as 0.8)

Output: a set of rules $\mathcal{R}$ that explains the occurrences of objects in the scenes $\mathcal{I}$

Choose the set of potential non-terminals as $\mathcal{V}' = \{X_i \in \mathcal{V} : deg_{out}(X_i)/(deg_{in}(X_i) + \epsilon) > 1\}$; `/* Proportion of the outward degree and inward degree;` $\epsilon < 1$ `*/`

Generate set of concepts and associated rules $\{\mathcal{R}_j\}_{j \in \mathcal{V}'}$ by choosing adjacent objects ; `/* Some examples are displayed in Figure 5 */`

Initialize $\mathcal{R} \leftarrow \emptyset$, $\mathcal{V}^\star = \emptyset$, and $C = \emptyset$; `/* Initialize by empty set; */`

while *the cover set* $\mathcal{C}$ *covers the dataset with probability* p **do**

  for *every non-terminal and associated set of rules* $\mathcal{R}_j$, $\forall j \in \mathcal{V}' \setminus \mathcal{V}^\star$ **do**

    Compute the gain $\mathcal{G}_{gain}(\mathcal{R}_j, \mathcal{R})$ as referred in Eq. 2 of the main draft

  compute next anchor node $X_{\bar{j}} = \arg\max_{X_j \in \mathcal{V}' \setminus \mathcal{V}^\star} \mathcal{G}_{gain}(\mathcal{R}_j, \mathcal{R})$ and $\mathcal{V}^\star = \mathcal{V}^\star \cup \bar{X}_j$

  $\mathcal{R} = \mathcal{R} \cup \mathcal{R}_{\bar{j}}$

  $C = C \cup C_{\bar{j}}$; `/* Update the rule set and the cover set; */`

$\mathcal{R} = \mathcal{R} \cup [S \rightarrow$ `'None'`$]$

return $\mathcal{R}$

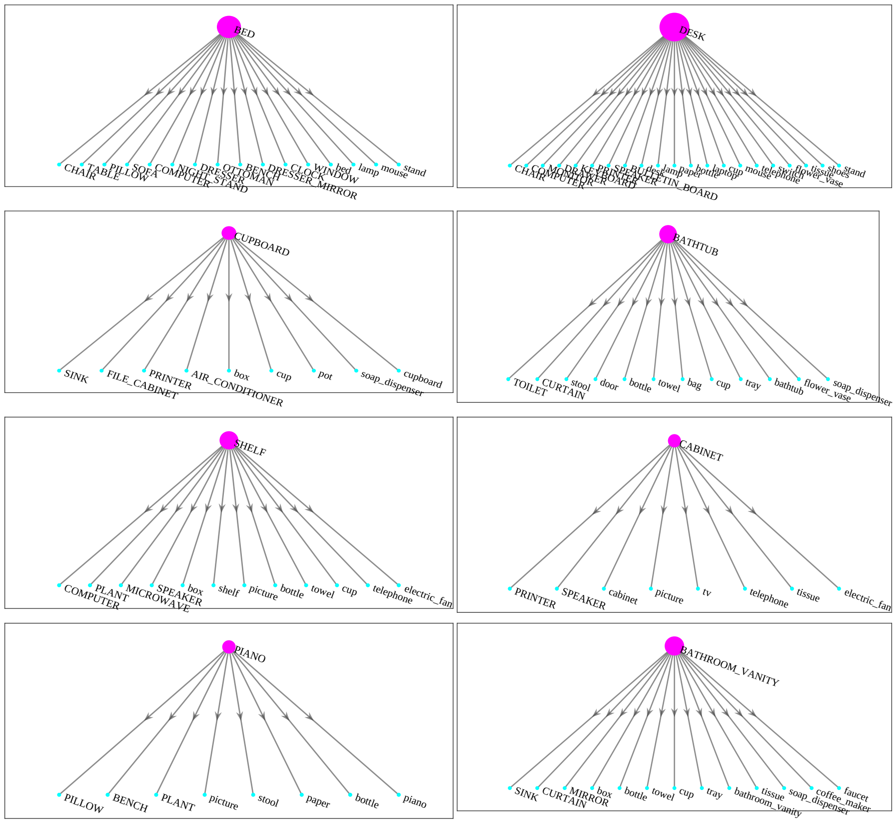


Fig. 5. A few examples of the production rule-sets R_j which are utilized to generate the CFG. The detail algorithm is furnished in algorithm 2. The non-terminal symbols are displayed in upper-case and terminal symbols are shown in lower-case.

5 Production rules extracted from SUN RGB-D

Here we describe the production rules of the CFG extracted from SUN RGB-D dataset in detail. The total number of rules generated by the algorithm described in section 3 of the main draft is 399, number of non-terminals is 49 and number of terminal objects is 84. In the following we display the entire learned grammar. Note again that the non-terminal symbols are displayed in upper case, S is the start symbol and *None* is the empty object. The rules are separated by semicolons ‘ ; ’ symbol.

```
  S → scene SCENE;
SCENE → counter COUNTER SCENE;
COUNTER → chair CHAIR COUNTER;
COUNTER → cabinet CABINET COUNTER;
COUNTER → computer COMPUTER COUNTER;
COUNTER → monitor MONITOR COUNTER;
COUNTER → sink SINK COUNTER;
COUNTER → drawer DRAWER COUNTER;
COUNTER → keyboard KEYBOARD COUNTER;
COUNTER → ottoman OTTOMAN COUNTER;
COUNTER → fridge FRIDGE COUNTER;
COUNTER → printer PRINTER COUNTER;
COUNTER → microwave MICROWAVE COUNTER;
COUNTER → mirror MIRROR COUNTER;
COUNTER → speaker SPEAKER COUNTER;
COUNTER → clock CLOCK COUNTER;
COUNTER → box COUNTER;
COUNTER → garbage_bin COUNTER;
COUNTER → lamp COUNTER;
COUNTER → stool COUNTER;
COUNTER → paper COUNTER;
COUNTER → tv COUNTER;
COUNTER → bottle COUNTER;
COUNTER → laptop COUNTER;
COUNTER → towel COUNTER;
COUNTER → bag COUNTER;
COUNTER → cup COUNTER;
COUNTER → tray COUNTER;
COUNTER → mouse COUNTER;
COUNTER → telephone COUNTER;
COUNTER → pot COUNTER;
COUNTER → flower_vase COUNTER;
COUNTER → coffee_maker COUNTER;
COUNTER → dishwasher COUNTER;
COUNTER → None;
SCENE → KITCHEN_COUNTER KITCHEN_COUNTER
SCENE;
KITCHEN_COUNTER → cabinet CABINET
KITCHEN_COUNTER;
KITCHEN_COUNTER → computer COMPUTER
KITCHEN_COUNTER;
KITCHEN_COUNTER → monitor MONITOR
KITCHEN_COUNTER;
KITCHEN_COUNTER → sink SINK KITCHEN_COUNTER;
KITCHEN_COUNTER → kitchen_cabinet
KITCHEN_CABINET KITCHEN_COUNTER;
KITCHEN_COUNTER → whiteboard WHITEBOARD
KITCHEN_COUNTER;
KITCHEN_COUNTER → fridge FRIDGE
KITCHEN_COUNTER;
KITCHEN_COUNTER → plant PLANT
KITCHEN_COUNTER;
KITCHEN_COUNTER → microwave MICROWAVE
KITCHEN_COUNTER;
KITCHEN_COUNTER → cart CART KITCHEN_COUNTER;
KITCHEN_COUNTER → box KITCHEN_COUNTER;
KITCHEN_COUNTER → garbage_bin
KITCHEN_COUNTER;
KITCHEN_COUNTER → stool KITCHEN_COUNTER;
KITCHEN_COUNTER → paper KITCHEN_COUNTER;
KITCHEN_COUNTER → tv KITCHEN_COUNTER;
KITCHEN_COUNTER → bottle KITCHEN_COUNTER;
KITCHEN_COUNTER → towel KITCHEN_COUNTER;
KITCHEN_COUNTER → bag KITCHEN_COUNTER;
KITCHEN_COUNTER → cup KITCHEN_COUNTER;
KITCHEN_COUNTER → tray KITCHEN_COUNTER;
KITCHEN_COUNTER → telephone KITCHEN_COUNTER;
KITCHEN_COUNTER → pot KITCHEN_COUNTER;
KITCHEN_COUNTER → soap_dispenser
KITCHEN_COUNTER;
KITCHEN_COUNTER → coffee_maker
KITCHEN_COUNTER;
KITCHEN_COUNTER → dishwasher
KITCHEN_COUNTER;
KITCHEN_COUNTER → faucet KITCHEN_COUNTER;
KITCHEN_COUNTER → None;
SCENE → bed BED SCENE;
BED → chair CHAIR BED;
BED → table TABLE BED;
BED → pillow PILLOW BED;
BED → sofa SOFA BED;
BED → computer COMPUTER BED;
BED → night_stand NIGHT_STAND BED;
BED → dresser DRESSER BED;
BED → ottoman OTTOMAN BED;
BED → bench BENCH BED;
BED → dresser_mirror dresser_MIRROR BED;
BED → clock CLOCK BED;
BED → window WINDOW BED;
BED → lamp BED;
BED → mouse BED;
BED → stand BED;
BED → None;
SCENE → desk DESK SCENE;
DESK → chair CHAIR DESK;
DESK → computer COMPUTER DESK;
DESK → monitor MONITOR DESK;
DESK → drawer DRAWER DESK;
DESK → keyboard KEYBOARD DESK;
DESK → printer PRINTER DESK;
DESK → speaker SPEAKER DESK;
DESK → bulletin_board BULLETIN_BOARD DESK;
DESK → lamp DESK;
DESK → paper DESK;
DESK → bottle DESK;
DESK → laptop DESK;
DESK → cup DESK;
DESK → mouse DESK;
DESK → telephone DESK;
DESK → switch DESK;
DESK → flower_vase DESK;
DESK → tissue DESK;
DESK → shoes DESK;
DESK → stand DESK;
DESK → None;
SCENE → bathroom_vanity BATHROOM_VANITY
SCENE;
BATHROOM_VANITY → sink SINK BATHROOM_VANITY;
BATHROOM_VANITY → curtain CURTAIN
BATHROOM_VANITY;
BATHROOM_VANITY → mirror MIRROR
BATHROOM_VANITY;
BATHROOM_VANITY → box BATHROOM_VANITY;
BATHROOM_VANITY → bottle BATHROOM_VANITY;
BATHROOM_VANITY → towel BATHROOM_VANITY;
BATHROOM_VANITY → cup BATHROOM_VANITY;
BATHROOM_VANITY → tray BATHROOM_VANITY;
BATHROOM_VANITY → tissue BATHROOM_VANITY;
BATHROOM_VANITY → soap_dispenser
BATHROOM_VANITY;
BATHROOM_VANITY → coffee_maker
BATHROOM_VANITY;
BATHROOM_VANITY → faucet BATHROOM_VANITY;
BATHROOM_VANITY → None;
SCENE → bathtub BATHTUB SCENE;
BATHTUB → toilet TOILET BATHTUB;
BATHTUB → curtain CURTAIN BATHTUB;
BATHTUB → stool BATHTUB;
BATHTUB → door BATHTUB;
BATHTUB → bottle BATHTUB;
BATHTUB → towel BATHTUB;
BATHTUB → bag BATHTUB;
BATHTUB → cup BATHTUB;
BATHTUB → tray BATHTUB;
BATHTUB → flower_vase BATHTUB;
BATHTUB → soap_dispenser BATHTUB;
BATHTUB → None;
SCENE → cupboard CUPBOARD SCENE;
CUPBOARD → sink SINK CUPBOARD;
CUPBOARD → file_cabinet FILE_CABINET
CUPBOARD;
CUPBOARD → printer PRINTER CUPBOARD;
CUPBOARD → air_conditioner AIR_CONDITIONER
CUPBOARD;
CUPBOARD → box CUPBOARD;
CUPBOARD → cup CUPBOARD;
CUPBOARD → pot CUPBOARD;
CUPBOARD → soap_dispenser CUPBOARD;
CUPBOARD → None;
SCENE → kitchen_cabinet KITCHEN_CABINET
SCENE;
KITCHEN_CABINET → sink SINK KITCHEN_CABINET;
KITCHEN_CABINET → fridge FRIDGE
KITCHEN_CABINET;
KITCHEN_CABINET → microwave MICROWAVE
KITCHEN_CABINET;
KITCHEN_CABINET → lamp KITCHEN_CABINET;
KITCHEN_CABINET → paper KITCHEN_CABINET;
KITCHEN_CABINET → bottle KITCHEN_CABINET;
KITCHEN_CABINET → bag KITCHEN_CABINET;
KITCHEN_CABINET → switch KITCHEN_CABINET;
KITCHEN_CABINET → soap_dispenser
KITCHEN_CABINET;
KITCHEN_CABINET → coffee_maker
KITCHEN_CABINET;
KITCHEN_CABINET → dishwasher
KITCHEN_CABINET;
KITCHEN_CABINET → faucet KITCHEN_CABINET;
KITCHEN_CABINET → None;
SCENE → table TABLE SCENE;
```

```
TABLE → chair CHAIR TABLE;
TABLE → computer COMPUTER TABLE;
TABLE → monitor MONITOR TABLE;
TABLE → keyboard KEYBOARD TABLE;
TABLE → printer PRINTER TABLE;
TABLE → speaker SPEAKER TABLE;
TABLE → lamp TABLE;
TABLE → stool TABLE;
TABLE → paper TABLE;
TABLE → bottle TABLE;
TABLE → laptop TABLE;
TABLE → towel TABLE;
TABLE → cup TABLE;
TABLE → tray TABLE;
TABLE → mouse TABLE;
TABLE → telephone TABLE;
TABLE → pot TABLE;
TABLE → flower_vase TABLE;
TABLE → tissue TABLE;
TABLE → coffee_maker TABLE;
TABLE → None;
SCENE → shelf SHELF SCENE;
SHELF → computer COMPUTER SHELF;
SHELF → plant PLANT SHELF;
SHELF → microwave MICROWAVE SHELF;
SHELF → speaker SPEAKER SHELF;
SHELF → box SHELF;
SHELF → picture SHELF;
SHELF → bottle SHELF;
SHELF → towel SHELF;
SHELF → cup SHELF;
SHELF → telephone SHELF;
SHELF → electric_fan SHELF;
SHELF → None;
SCENE → piano PIANO SCENE;
PIANO → pillow PILLOW PIANO;
PIANO → bench BENCH PIANO;
PIANO → plant PLANT PIANO;
PIANO → picture PIANO;
PIANO → stool PIANO;
PIANO → paper PIANO;
PIANO → bottle PIANO;
PIANO → None;
SCENE → drawer DRAWER SCENE;
DRAWER → sink SINK DRAWER;
DRAWER → printer PRINTER DRAWER;
DRAWER → speaker SPEAKER DRAWER;
DRAWER → tv DRAWER;
DRAWER → bottle DRAWER;
DRAWER → tray DRAWER;
DRAWER → telephone DRAWER;
DRAWER → flower_vase DRAWER;
DRAWER → toilet_paper DRAWER;
DRAWER → electric_fan DRAWER;
DRAWER → coffee_maker DRAWER;
DRAWER → None;
SCENE → dresser DRESSER SCENE;
DRESSER → pillow PILLOW DRESSER;
DRESSER → night_stand NIGHT_STAND DRESSER;
DRESSER → box DRESSER;
DRESSER → lamp DRESSER;
DRESSER → tv DRESSER;
DRESSER → bottle DRESSER;
DRESSER → laptop DRESSER;
DRESSER → cup DRESSER;
DRESSER → None;
SCENE → sofa SOFA SCENE;
SOFA → pillow PILLOW SOFA;
SOFA → sofa_chair SOFA_CHAIR SOFA;
SOFA → endtable ENDTABLE SOFA;
SOFA → COFFEE_TABLE COFFEE_TABLE SOFA;
SOFA → cart CART SOFA;
SOFA → towel SOFA;
SOFA → None;
SCENE → night_stand NIGHT_STAND SCENE;
NIGHT_STAND → box NIGHT_STAND;
NIGHT_STAND → garbage_bin NIGHT_STAND;
NIGHT_STAND → lamp NIGHT_STAND;
NIGHT_STAND → bottle NIGHT_STAND;
NIGHT_STAND → telephone NIGHT_STAND;
NIGHT_STAND → None;
SCENE → BOOKSHELF BOOKSHELF SCENE;
BOOKSHELF → bulletin_board BULLETIN_BOARD
BOOKSHELF;
BOOKSHELF → stool BOOKSHELF;
BOOKSHELF → paper BOOKSHELF;
BOOKSHELF → bottle BOOKSHELF;
BOOKSHELF → cup BOOKSHELF;
BOOKSHELF → None;
SCENE → COFFEE_TABLE COFFEE_TABLE SCENE;
COFFEE_TABLE → pillow PILLOW COFFEE_TABLE;
COFFEE_TABLE → sofa_chair SOFA_CHAIR
COFFEE_TABLE;
COFFEE_TABLE → endtable ENDTABLE
COFFEE_TABLE;
COFFEE_TABLE → paper COFFEE_TABLE;
COFFEE_TABLE → cup COFFEE_TABLE;
COFFEE_TABLE → tray COFFEE_TABLE;
COFFEE_TABLE → None;
SCENE → rack RACK SCENE;
RACK → printer PRINTER RACK;
RACK → towel RACK;
RACK → bag RACK;
RACK → None;
SCENE → dining_table DINING_TABLE SCENE;
DINING_TABLE → chair CHAIR DINING_TABLE;
DINING_TABLE → painting PAINTING
DINING_TABLE;
DINING_TABLE → ottoman OTTOMAN DINING_TABLE;
DINING_TABLE → laptop DINING_TABLE;
DINING_TABLE → None;
SCENE → window WINDOW SCENE;
WINDOW → pillow PILLOW WINDOW;
WINDOW → plant PLANT WINDOW;
WINDOW → towel WINDOW;
WINDOW → None;
SCENE → endtable ENDTABLE SCENE;
ENDTABLE → microwave MICROWAVE ENDTABLE;
ENDTABLE → speaker SPEAKER ENDTABLE;
ENDTABLE → box ENDTABLE;
ENDTABLE → lamp ENDTABLE;
ENDTABLE → bottle ENDTABLE;
ENDTABLE → cup ENDTABLE;
ENDTABLE → tray ENDTABLE;
ENDTABLE → flower_vase ENDTABLE;
ENDTABLE → electric_fan ENDTABLE;
ENDTABLE → None;
SCENE → cabinet CABINET SCENE;
CABINET → printer PRINTER CABINET;
CABINET → speaker SPEAKER CABINET;
CABINET → picture CABINET;
CABINET → tv CABINET;
CABINET → telephone CABINET;
CABINET → tissue CABINET;
CABINET → electric_fan CABINET;
CABINET → None;
SCENE → dresser_mirror dresser_MIRROR SCENE;
dresser_MIRROR → tv dresser_MIRROR;
dresser_MIRROR → None;
SCENE → fridge FRIDGE SCENE;
FRIDGE → plant PLANT FRIDGE;
FRIDGE → microwave MICROWAVE FRIDGE;
FRIDGE → clock CLOCK FRIDGE;
FRIDGE → box FRIDGE;
FRIDGE → bottle FRIDGE;
FRIDGE → soap_dispenser FRIDGE;
FRIDGE → None;
SCENE → person PERSON SCENE;
PERSON → shoes PERSON;
PERSON → None;
SCENE → cart CART SCENE;
CART → keyboard KEYBOARD CART;
CART → printer PRINTER CART;
CART → bench BENCH CART;
CART → speaker SPEAKER CART;
CART → tv CART;
CART → mouse CART;
CART → None;
SCENE → file_cabinet FILE_CABINET SCENE;
FILE_CABINET → plant PLANT FILE_CABINET;
FILE_CABINET → microwave MICROWAVE
FILE_CABINET;
FILE_CABINET → flower_vase FILE_CABINET;
FILE_CABINET → electric_fan FILE_CABINET;
FILE_CABINET → None;
SCENE → toilet TOILET SCENE;
TOILET → sink SINK TOILET;
TOILET → garbage_bin TOILET;
TOILET → paper TOILET;
TOILET → tissue TOILET;
TOILET → toilet_paper_dispenser TOILET;
TOILET → None;
SCENE → bulletin_board BULLETIN_BOARD SCENE;
BULLETIN_BOARD → clock CLOCK BULLETIN_BOARD;
BULLETIN_BOARD → laptop BULLETIN_BOARD;
BULLETIN_BOARD → ladder BULLETIN_BOARD;
BULLETIN_BOARD → None;
SCENE → bench BENCH SCENE;
BENCH → pillow PILLOW BENCH;
BENCH → podium BENCH;
```

```
BENCH → None;
SCENE → tv_stand TV_STAND SCENE;
TV_STAND → paper TV_STAND;
TV_STAND → tv TV_STAND;
TV_STAND → None;
SCENE → clock CLOCK SCENE;
CLOCK → pillow PILLOW CLOCK;
CLOCK → sofa_chair SOFA_CHAIR CLOCK;
CLOCK → ottoman OTTOMAN CLOCK;
CLOCK → switch CLOCK;
CLOCK → None;
SCENE → sink SINK SCENE;
SINK → microwave MICROWAVE SINK;
SINK → mirror MIRROR SINK;
SINK → bottle SINK;
SINK → towel SINK;
SINK → soap_dispenser SINK;
SINK → faucet SINK;
SINK → None;
SCENE → air_conditioner AIR_CONDITIONER
SCENE;
AIR_CONDITIONER → curtain CURTAIN
AIR_CONDITIONER;
AIR_CONDITIONER → ladder AIR_CONDITIONER;
AIR_CONDITIONER → None;
SCENE → sofa_chair SOFA_CHAIR SCENE;
SOFA_CHAIR → chair CHAIR SOFA_CHAIR;
SOFA_CHAIR → None;
SCENE → mirror MIRROR SCENE;
MIRROR → faucet MIRROR;
MIRROR → None;
SCENE → painting PAINTING SCENE;
PAINTING → switch PAINTING;
PAINTING → None;
SCENE → printer PRINTER SCENE;
PRINTER → tissue PRINTER;
PRINTER → None;
SCENE → plant PLANT SCENE;
PLANT → pot PLANT;
PLANT → electric_fan PLANT;
PLANT → None;
SCENE → curtain CURTAIN SCENE;
CURTAIN → garbage_bin CURTAIN;
CURTAIN → None;
SCENE → pillow PILLOW SCENE;
PILLOW → lamp PILLOW;
PILLOW → None;
SCENE → computer COMPUTER SCENE;
COMPUTER → keyboard KEYBOARD COMPUTER;
COMPUTER → mouse COMPUTER;
COMPUTER → None;
SCENE → whiteboard WHITEBOARD SCENE;
WHITEBOARD → picture WHITEBOARD;
WHITEBOARD → None;
SCENE → ottoman OTTOMAN SCENE;
OTTOMAN → cup OTTOMAN;
OTTOMAN → None;
SCENE → speaker SPEAKER SCENE;
SPEAKER → cup SPEAKER;
SPEAKER → None;
SCENE → monitor MONITOR SCENE;
MONITOR → keyboard KEYBOARD MONITOR;
MONITOR → None;
SCENE → chair CHAIR SCENE;
CHAIR → towel CHAIR;
CHAIR → None;
SCENE → keyboard KEYBOARD SCENE;
KEYBOARD → mouse KEYBOARD;
KEYBOARD → None;
SCENE → microwave MICROWAVE SCENE;
MICROWAVE → pot MICROWAVE;
MICROWAVE → None;
SCENE → None
```

References

1. Dorit Dor and Michael Tarsi. A simple algorithm to construct a consistent extension of a partially oriented graph. 1992. 9
2. Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018. 8
3. Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. In *Proceedings of ICML*, pages 1945–1954. JMLR. org, 2017. 8
4. 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):12, 2019. 4, 5
5. Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008. 3
6. Judea Pearl and Thomas S Verma. A theory of inferred causation. In *Studies in Logic and the Foundations of Mathematics*, volume 134, pages 789–811. Elsevier, 1995. 9
7. Siyuan Qi, Yixin Zhu, Siyuan Huang, Chenfanfu Jiang, and Song-Chun Zhu. Human-centric indoor scene synthesis using stochastic grammar. In *Proceedings of CVPR*, pages 5899–5908, 2018. 4, 5
8. Daniel Ritchie, Kai Wang, and Yu-an Lin. Fast and flexible indoor scene synthesis via deep convolutional generative models. In *Proceedings of CVPR*, pages 6182–6190, 2019. 4, 5
9. Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of CVPR*, pages 567–576, 2015. 2
10. Shuran Song and Jianxiong Xiao. Deep sliding shapes for amodal 3d object detection in rgb-d images. In *Proceedings of CVPR*, pages 808–816, 2016. 7
11. Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *Proceedings of CVPR*, pages 1746–1754, 2017.
12. Lap-Fai Yu, Sai-Kit Yeung, Chi-Keung Tang, Demetri Terzopoulos, Tony F Chan, and Stanley J Osher. Make it home: automatic optimization of furniture arrangement. In *ACM Transactions on Graphics (TOG)*, volume 30, page 86. ACM, 2011. 8
13. Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Proceedings of NIPS*, pages 487–495, 2014. 4, 7