Supplementary Material: Scene Grammar
Variational Autoencoder

Pulak Purkait\textsuperscript{1}[0000–0003–0684–1209], Christopher Zach\textsuperscript{2}[0000–0003–2840–6187],
and Ian Reid\textsuperscript{1}[0000–0001–7790–6423]

\textsuperscript{1} Australian Institute of Machine Learning and School of Computer Science,
The University of Adelaide, Adelaide SA 5005, Australia
\textsuperscript{2} Chalmers University of Technology, Goteborg 41296, Sweden

Table of Contents

1 Selection of the CFG—Algorithm details . . . . . . . . . . . . . . 1
2 Visualization of the latent space . . . . . . . . . . . . . . . . . 3
3 Results on SUNCG . . . . . . . . . . . . . . . . . . . . . . . . . . 4
4 Additional experiments on SUN RGBD . . . . . . . . . . . . . . 4
4.1 Scene layout from the RGB-D image—in detail . . . . . 4
4.2 Quality assessment of the autoencoder . . . . . . . . . . . 7
5 Production rules extracted from SUN RGB-D . . . . . . . . . . 10

List of Figures

1 2D projection of the mean $\mu$ using t-SNE . . . . . . . . . . . . . . 3
2 Sampled views by baseline methods on the SUNCG datasets . . . 5
3 1-1 comparison with Grains on SUNCG datasets . . . . . . . . . 5
4 Latent code interpolation on SUNCG . . . . . . . . . . . . . . . . 6
5 Examples of the production rule-sets $R_j$ . . . . . . . . . . . . . 10

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 $R_j$ be the set of production rules derived from $X_j$ in
the causal graph. Let $C_j$ be the set of terminals that $R_j$ covers (essentially
nodes that $X_j$ leads to in the causal graph $G$). Our greedy algorithm begins
with an empty set $R = \emptyset$ and chooses the node $X_j$ and associated production
rule set $R_j$ to add that maximize the gain in coverage

$$G_{gain}(R_j, R) = \frac{1}{|R_j|} \sum_{I_i \in \bigcup R_j \cap C} |Y_i| \big/ |I_i|$$

(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 \( \Sigma \). The set of all possible objects including dummy \texttt{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 \( X_j \text{inc} \) 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 \( |E| \) 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:

\[
\begin{align*}
S & \rightarrow \text{scene \texttt{SCENE}} \\
\text{SCENE} & \rightarrow \text{bed \texttt{BED} \texttt{SCENE}} \\
\text{BED} & \rightarrow \text{bed \texttt{BED}} \\
\text{BED} & \rightarrow \text{lamp \texttt{BED}} \\
\text{Bed} & \rightarrow \text{sofa \texttt{SOFA} \texttt{BED}} \\
\text{BED} & \rightarrow \text{pillow \texttt{PILLOW} \texttt{SOFA}} \\
\text{BED} & \rightarrow \text{night_stand \texttt{BED}} \\
\text{BED} & \rightarrow \text{dresser \texttt{BED}} \\
\text{BED} & \rightarrow \text{None} \\
\text{SCENE} & \rightarrow \text{sofainc \texttt{SCENE}} \\
\text{SOFA} & \rightarrow \text{sofa \texttt{SOFA}} \\
\text{SOFA} & \rightarrow \text{pillow \texttt{PILLOW} \texttt{SOFA}} \\
\text{SOFA} & \rightarrow \text{TOWEL \texttt{SOFA}} \\
\text{SOFA} & \rightarrow \text{None} \\
\text{SCENE} & \rightarrow \text{None}
\end{align*}
\]

The entire grammar is displayed in Section 5. In the above snippet the non-terminal \texttt{BED} generates another non-terminal \texttt{SOFA} that leads to an additional set of production rules corresponding to \texttt{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 \textbf{BED} generates another non-terminal \textbf{SOFA} which further leads to another set of production rules corresponding to \textbf{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 (\(i.e., \text{the mean } \mu \text{ of the distribution } \mathcal{N}(\mu, \Sigma)\)) 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 \textit{chair} and a \textit{table} are mapped to two nearby but distinct points.

![Fig. 1. We plot the 2D projection of the mean \( \mu \) 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.](image-url)
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, 14,814 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 $\alpha \mu_1 + (1-\alpha) \mu_2$ of the means $\mu_1$ and $\mu_2$ 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 $F_i$ be the (in our case 8192-dimensional) deep feature vector extracted from the image $I_i$, and let $N(\mu_i, \Sigma_i)$ be the (e.g. 50-dimensional) latent representation obtained by encoding the corresponding parse tree. We align the feature vector $F_i$ 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.
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.

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 $\alpha \mu_1 + (1 - \alpha) \mu_2$ of the means $\mu_1$ and $\mu_2$ 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.
\[ \psi(F_i) = AF_i, \text{ where } A \text{ is a matrix to be learned from a training set } \mathcal{T} = \{ I_i := (F_i, \mu_i, \Sigma_i) \}. \]

We minimize the cross-entropy between the predicted (deterministic) latent representation \( \psi(F_i) \) and the target distribution \( N(\mu_i, \Sigma_i) \), therefore the optimal matrix \( A \) is determined as

\[
\hat{A} = \arg \min_A \sum_{I_i \in \mathcal{T}} (AF_i - \mu_i)^T \Sigma_i^{-1} (AF_i - \mu_i) + \lambda\|A\|^2_2.
\]

The features \( F_i \) of dimension 8192 are then projected into the mean of the encoded vector \( \mu_i \) (typically dimension 50). Let \( \phi : F_i \rightarrow \mu_i \) be the mapping that project the feature vectors \( F_i \) to the latent space \( \mu_i \). A neural network could be used to learn the mapping \( \phi \), however, a simple linear projection is employed here, i.e., \( \psi(F_i) = AF_i \).

Differentiating the objective to zero, we get

\[
\hat{A} = \sum_{I_i \in \mathcal{T}} F_i^T \Sigma_i^{-1} F_i + 2\lambda I \]

\[
\hat{A}^{-1} \sum_{I_i \in \mathcal{Tr}} F_i \Sigma_i^{-1} \mu_i
\]

where we also added a regularization term with weight \( \lambda \) (chosen as \( \lambda = 100 \)).

\section{Testing}

For test data image features \cite{13} are extracted and then mapped to the latent space using the trained mapping \( \hat{\phi}(F_i) := F_i \hat{A} \). The scenes are then decoded from the latent vectors \( \hat{\mu} := F_i \hat{A} \) using the decoder part of the SG-VAE. The bounding box detector of DSS \cite{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.

\subsection{Quality assessment of the autoencoder}

The scenes and the bounding boxes of the test examples are first encoded to the latent representations \( N(\mu_i, \Sigma_i) \). The mean of the distributions \( \mu_i \) 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:
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 $|R| \times \theta$ dimensional vector where $|R|$ is the number of production rules and $\theta$ is the size of the attributes. For example, the pose and shape attributes $\Theta_j \rightarrow^k = (P_j \rightarrow^k, S_k^i)$, associated with a production rule (say $p$th rule) in which a non-terminal $X_j$ yields a terminal $X_k$, are placed in $(p - 1) \times \theta + 1 : p \times \theta$ 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 $\theta$ dimensional vector only.

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 $|V|$ dimensional 1-hot vector and a $\theta$ 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.

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 \( D \) of natural scenes formed by a set of objects
\[ \mathcal{V} = \{X_1, \ldots, X_n\} \]

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

1. Initialize \( G := (\mathcal{V}, \mathcal{E}_0) \); /* Initialize with prior edges */
2. for every pair of objects \((X_j, X_j') \in \mathcal{V} \) do
   3. for every conditioning variable \( X_k \in \mathcal{V} \setminus \{X_i, X_i'\} \) do
   4. hypothesis test \( X_j \perp \perp X_j' \mid X_k \) in \( D \); /* Using Conditional Algo 1 */
   5. if no independence was found then
      6. add an undirected edge \((X_j, X_j')\) in \( \mathcal{E} \), i.e., \( \mathcal{E} = \mathcal{E} \cup \{X_j, X_j'\} \).
   7. for every pair of objects \((X_j, X_j') \in \mathcal{V} \) with a common neighbor \( X_k \) do
      8. if \((X_j, X_j') \notin \mathcal{E}\) then
         9. if one of \((X_k, X_j)\) and \((X_k, X_j')\) is directed and the other is undirected OR both are undirected then
            10. turn the triplet into a common parent structure, i.e., \( X_j \rightarrow X_k \rightarrow X_j' \).
      12. Propagate the arrow orientation for all undirected edges (modify the set \( \mathcal{E} \)) accordingly without introducing a directed cycle.; /* Following Dor and Tarsi [1] */
   13. return \( G = (\mathcal{V}, \mathcal{E}) \)

Algorithm 2: \( p \)-Cover : A greedy algorithm for \( p \)-cover

Input: a dataset \( D \) of indoor scenes \( \mathcal{I} \) formed by a set of objects
\[ \mathcal{V} = \{X_1, \ldots, X_n\} \], the causal graph \( G = (\mathcal{V}, \mathcal{E}) \) 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
\[ \mathcal{V}' = \{X_i \in \mathcal{V} : \text{deg}_{\text{out}}(X_i)/(\text{deg}_{\text{in}}(X_i) + \epsilon) > 1\} \]; /* Proportion of the outward degree and inward degree; \( \epsilon < 1 \) */

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 */
2. Initialize \( \mathcal{R} = \emptyset \), \( \mathcal{V}' = \emptyset \), and \( C = \emptyset \); /* Initialize by empty set */
3. while the cover set \( \mathcal{C} \) covers the dataset with probability \( p \) do
   4. for every non-terminal and associated set of rules \( \mathcal{R}_j \), \( \forall j \in \mathcal{V}' \) do
      5. Compute the gain \( \mathcal{G}_{\text{gain}}(\mathcal{R}_j, \mathcal{R}) \) as referred in Eq. 2 of the main draft
      6. compute next anchor node \( X_j = \arg\max_{X_j \in \mathcal{V}' \setminus \mathcal{V}'} \mathcal{G}_{\text{gain}}(\mathcal{R}_j, \mathcal{R}) \) and
         \( \mathcal{V}' = \mathcal{V}' \cup X_j \)
      7. \( \mathcal{R} = \mathcal{R} \cup \mathcal{R}_j \), \( C = C \cup C_j \); /* Update the rule set and the cover set */
   10. \( \mathcal{R} = \mathcal{R} \cup [S \rightarrow 'None'] \)
11. return \( \mathcal{R} \)
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 → cabinet CABINET COUNTER;
COUNTER → computer COMPUTER COUNTER;
COUNTER → drawer DRAWER COUNTER;
COUNTER → mirror MIRROR COUNTER;
COUNTER → ottoman OTTOMAN COUNTER;
COUNTER → fridge FRIDGE COUNTER;
COUNTER → printプリンター COUNTER;
COUNTER → microwave MICROWAVE COUNTER;
COUNTER → sink SINK COUNTER;
COUNTER → speaker SPEAKER COUNTER;
COUNTER → towel TOWEL COUNTER;
COUNTER → toilet TOILET COUNTER;
COUNTER → telephone TELEPHONE COUNTER;
COUNTER → towel TOWEL COUNTER;
COUNTER → table TABLE COUNTER;
COUNTER → refrigerator FRIDGE;
COUNTER → bed BED COUNTER;
COUNTER → cabinet CABINET COUNTER;
COUNTER → chair CHAIR COUNTER;
COUNTER → desk DESK COUNTER;
COUNTER → door DOOR COUNTER;
COUNTER → drawer DRAWER COUNTER;
COUNTER → floor FLOOR COUNTER;
COUNTER → sink SINK COUNTER;
COUNTER → stove STOVE COUNTER;
COUNTER → towel TOWEL COUNTER;
COUNTER → ottoman OTTOMAN COUNTER;
COUNTER → speaker SPEAKER COUNTER;
COUNTER → toilet TOILET COUNTER;
COUNTER → bed BED COUNTER;
COUNTER → cabinet CABINET COUNTER;
COUNTER → chair CHAIR COUNTER;
COUNTER → desk DESK COUNTER;
COUNTER → drawer DRAWER COUNTER;
COUNTER → floor FLOOR COUNTER;
COUNTER → kitchen KITCHEN COUNTER;
COUNTER → sink SINK COUNTER;
COUNTER → table TABLE COUNTER;
COUNTER → refrigerator FRIDGE;
COUNTER → bed BED COUNTER;
COUNTER → cabinet CABINET COUNTER;
COUNTER → chair CHAIR COUNTER;
COUNTER → desk DESK COUNTER;
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 → 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


