Learning Canonical Representations for Scene Graph to Image Generation

Roei Herzig[1(✉)], Amir Bar[1], Huijuan Xu[2], Gal Chechik[3], Trevor Darrell[2], and Amir Globerson[1]

[1] Tel Aviv University, Tel Aviv-Yafo, Israel
[2] UC Berkeley, Berkeley, USA
[3] Bar-Ilan University, NVIDIA Research, Ramat Gan, Israel

Abstract. Generating realistic images of complex visual scenes becomes challenging when one wishes to control the structure of the generated images. Previous approaches showed that scenes with few entities can be controlled using scene graphs, but this approach struggles as the complexity of the graph (the number of objects and edges) increases. In this work, we show that one limitation of current methods is their inability to capture semantic equivalence in graphs. We present a novel model that addresses these issues by learning canonical graph representations from the data, resulting in improved image generation for complex visual scenes (The project page is available at https://roeiherz.github.io/CanonicalSg2Im/). Our model demonstrates improved empirical performance on large scene graphs, robustness to noise in the input scene graph, and generalization on semantically equivalent graphs. Finally, we show improved performance of the model on three different benchmarks: Visual Genome, COCO, and CLEVR.

Keywords: Scene graphs · Canonical representations · Image generation

1 Introduction

Generating realistic images is a key task in computer vision research. Recently, a series of methods were presented for creating realistic-looking images of objects and faces (e.g. [3,20,37]). Despite this impressive progress, a key challenge remains: how can one control the content of images at multiple levels to generate images that have specific desired composition and attributes. Controlling content can be particularly challenging when generating visual scenes that contain multiple interacting objects. One natural way of describing such scenes is via the structure of a *Scene Graph* (SG), which contains a set of objects as nodes and their attributes and relations as edges. Indeed, several studies addressed generating images from SGs [1,17,25]. Unfortunately, the quality of images generated

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-58574-7_13) contains supplementary material, which is available to authorized users.

A. Vedaldi et al. (Eds.): ECCV 2020, LNCS 12371, pp. 210–227, 2020.
https://doi.org/10.1007/978-3-030-58574-7_13

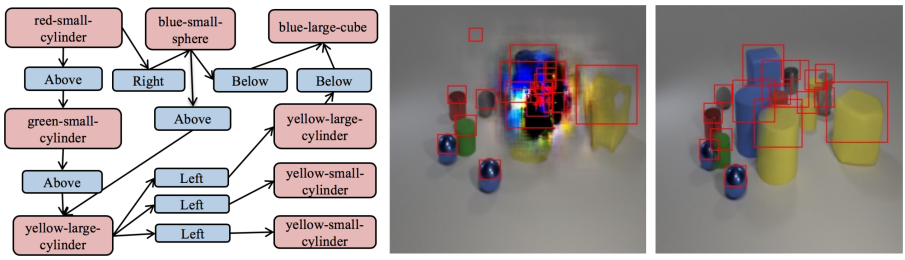


Fig. 1. Generation of scenes with many objects. Our method achieves better performance on such scenes than previous methods. **Left:** A partial input scene graph. **Middle:** Generation using [17]. **Right:** Generation using our proposed method.

from SGs still lags far behind that of generating single objects or faces. Here we show that one problem with current models is their failure to capture logical equivalences, and we propose an approach for overcoming this limitation.

SG-to-image typically involves two steps: first, generating a layout from the SG, and then generating pixels from the layout. In the first step, the SG does not contain bounding boxes, and is used to generate a layout that contains bounding box coordinates for all objects. The transformation relies on geometric properties specified in the SG such as "(A, right, B)". Since SGs are typically generated by humans, they usually do not contain *all* correct relations in the data. For example, in an SG with relation (A, right, B) it is always true that (B, left, A), yet typically only one of these relations will appear.[1] This example illustrates that multiple SGs can describe the same physical configuration, and are thus logically equivalent. Ideally, we would like all such SGs to result in the same layout and image. As we show here, this often does not hold for existing models, resulting in low-quality generated images for large graphs (see Fig. 1).

Here we present an approach to overcome the above difficulty. We first formalize the problem as being invariant to certain logical equivalences (i.e., all equivalent SGs should generate the same image). Next, we propose to replace any SG with a "canonical SG" such that all logically equivalent SGs are replaced by the same canonical SG, and this canonical SG is the one used in the layout generation step. This approach, by definition, results in the same output for all logically equivalent graphs. We present a practical approach to learning such a canonicalization process that does not use any prior knowledge about the relations (e.g., it does not know that "right" is a transitive relation). We show how to integrate the resulting canonical SGs within a SG-to-image generation model, and how to learn it from data. Our method also learns more compact models than previous methods, because the canonicalization process distributes information across the graph with only few additional parameters.

In summary, our novel contributions are as follows: 1) We propose a model that uses canonical representations of SGs, thus obtaining stronger invariance properties. This in turn leads to generalization on semantically equivalent graphs

[1] We note that human raters don't typically include all logically equivalent relations. We analyzed data and found only small fraction of these are annotated in practice.

and improved robustness to graph size and noise in comparison to existing methods. 2) We show how to learn the canonicalization process from data. 3) We use our canonical representations within an SG-to-image model and show that our approach results in improved generation on Visual Genome, COCO, and CLEVR, compared to the state-of-the-art baselines.

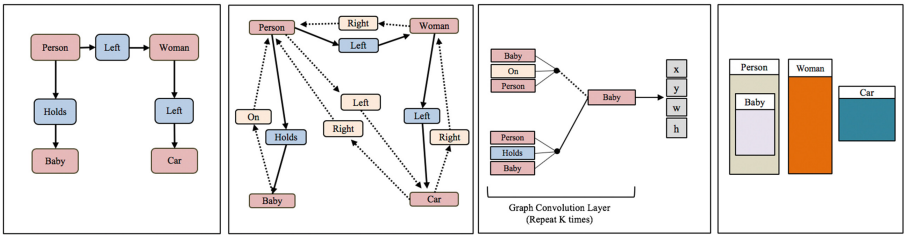
2 Related Work

Image generation. Earlier work on image generation used autoregressive networks [35,36] to model pixel conditional distributions. Recently, GANs [11] and VAEs [21] emerged as models of choice for this task. Specifically, generation techniques based on GANs were proposed for generating sharper, more diverse and better realistic images in a series of works [5,20,26,28,32,40,44,53,60,64].

Conditional image synthesis. Multiple works have explored approaches for generating images with a given desired content. Conditioning inputs may include class labels [7,30,34], source images [15,16,27,50,66,67], model interventions [2], and text [14,38,41,42,47,57,58,61]. Other studies [9,33] focused on image manipulation using language descriptions while disentangling the semantics of both input images and text descriptions.

Structured representation. Recent models [14,65] incorporate intermediate structured representations, such as layouts or skeletons, to control the coarse structure of generated images. Several studies focused on generating images from such representations (e.g., semantic segmentation masks [6,16,37,53], layout [62], and SGs [1,17,25]). Layout and SGs are more compact representations as compared to segmentation masks. While layout [62] provides spatial information, SGs [17] provide richer information about attributes and relations. Another advantage of SGs is that they are closely related to the semantics of the image as perceived by humans, and therefore editing an SG corresponds to clear changes in semantics. SGs and visual relations have also been used in image retrieval [19,46], relationship modeling [23,39,45], image captioning [56] and action recognition [12,29]. Several works have addressed the problem of generating SGs from text [46,51], standalone objects [55] and images [13].

Scene-graph-to-image generation. Sg2Im [17] was the first to propose an end-to-end method for generating images from scene graphs. However, as we note above, the current SG-to-image models [1,8,25,31,52] show degraded performance on complex SGs with many objects. To mitigate this, the authors in [1] have utilized stronger supervision in the form of a coarse grid, where attributes of location and size are specified for each object. The focus of our work is to alleviate this difficulty by directly modeling some of the invariances in SG representation. Finally, the topic of invariance in deep architectures has also attracted considerable interest, but mostly in the context of certain permutation invariances [13,59]. Our approach focuses on a more complex notion of invariance, and addresses it via canonicalization.



(a) Input Scene Graph (b) Weighted Scene Graph Canonicalization (c) GCN for Weighted Scene Graph (d) Generated Scene Layout

Fig. 2. Proposed Scene Graph to Layout architecture. (a) An input scene graph. (b) The graph is first canonicalized using our WSGC method in Sect. 3.2. Dashed edges correspond to completed relations that are assigned with weights. (c) A GCN is applied to the weighted graph, resulting in bounding box coordinates. (d) The GCN outputs are used to generate the predicted layout.

3 Scene Graph Canonicalization

As mentioned above, the same image can be represented by multiple logically-equivalent SGs. Next we define this formally and propose an approach to canonicalize graphs that enforces invariance to these equivalences. In Sect. 4 we show how to use this canonical scene graph within an SG-to-image task.

Let $\mathcal{C}$ be the set of objects categories and $\mathcal{R}$ be the set of possible relations.[2] An SG over n objects is a tuple (O, E) where $O \in \mathcal{C}^n$ is the object categories and E is a set of labeled directed edges (triplets) of the form (i, r, j) where $i, j \in \{1, \ldots, n\}$ and $r \in \mathcal{R}$. Thus an edge (i, r, j) implies that the i^{th} object (that has category o_i) should have relation r with the j^{th} object. Alternatively the set E can be viewed as a set of $|\mathcal{R}|$ directed graphs where for each r the graph E_r contains only the edges for relation r.

Our key observation is that relations in SGs are often dependent, because they reflect properties of the physical world. This means that for a relation r, the presence of certain edges in E_r implies that other edges have to hold. For example, assume r is a **transitive relation** like "left". Then if $i, j \in E_r$ and $j, k \in E_r$, it should hold that $i, k \in E_r$. There are also dependencies between different relations. For example, if r, r' are **converse relations** (e.g., r is "left" and r' "right") then $i, j \in E_r$ implies $j, i \in E_{r'}$. Formally, all the above dependencies are first order logic formulas. For example, r, r' being converse corresponds to the formula $\forall i, j : r(i, j) \implies r'(j, i)$. Let $\mathcal{F}$ denote this set of formulas.

The fact that certain relations are implied by a graph does not mean that they are contained in its set of relations. For example, E may contain $(1, \text{left}, 2)$ but not $(2, \text{right}, 1)$.[3] However, we would like SGs that contain either or both of these relations to result in the same image. In other words, we would like all logically equivalent graphs to result in the same image, as formally stated next.

[2] Objects in SGs also contain attributes but we drop these for notational simplicity.

[3] This is because empirical graphs E are created by human annotators, who typically skip redundant edges that can be inferred from other edges.

Given a scene graph E denote by $Q(E)$ the set of graphs that are logically equivalent to E.[4] As mentioned above, we would like all these graphs to result in the same image. Currently, SG-to-layout architectures do not have this invariance property because they operate on E and thus sensitive to whether it has certain edges or not. A natural approach to solve this is to replace E with a *canonical form* $C(E)$ such that $\forall E' \in Q(E)$ we have $C(E') = C(E)$. There are several ways of defining $C(E)$. Perhaps the most natural one is the "relation-closure" which is the graph containing all relations implied by those in E.

Definition 1. *Given a set of formulas $\mathcal{F}$, and relations E, the closure $C(E)$ is the set of relations that are true in any SG that contains E and satisfies $\mathcal{F}$.*

We note that the above definition coincides with the standard definition for closure of relations. Our definition emphasizes the fact that $C(E)$ are relations that are necessarily true given those in E. Additionally we allow for multiple relations, whereas closure is typically defined with respect to a single property. Next we describe how to calculate $C(E)$ when $\mathcal{F}$ is known, and then explain how to learn $\mathcal{F}$ from data.

3.1 Calculating Scene Graph Canonicalization

For a general set of formulas, calculating the closure is hard as it is an instance of inference in first order logic. However, here we restrict ourselves to the following formulas for which this calculation is efficient:[5]

- Transitive Relations: We assume a set of relations $\mathcal{R}_{trans} \subset \mathcal{R}$ where all $r \in \mathcal{R}_{trans}$ satisfy the formula $\forall x, y, z : r(x, y) \wedge r(y, z) \implies r(x, z)$.
- Converse Relations: We assume a set of relations pairs $\mathcal{R}_{conv} \subset \mathcal{R} \times \mathcal{R}$ where all $(r, r') \in \mathcal{R}_{conv}$ satisfy the formula $\forall x, y : r(x, y) \implies r'(y, x)$.

Under the above set of formulas, the closure $C(E)$ can be computed via the following procedure, which we call **Scene Graph Canonicalization (SGC)**:
Initialization: Set $C(E) = E$.
Converse Completion: $\forall (r, r') \in \mathcal{R}_{conv}$, if $(i, r, j) \in E$, add (j, r', i) to $C(E)$.
Transitive Completion: For each $r \in \mathcal{R}_{trans}$ calculate the transitive closure of $C_r(E)$ (namely the r relations in $C(E)$) and add it to $C(E)$. The transitive closure can be calculated using the Floyd-Warshall algorithm [10].

It can be shown (see Supplementary) that the SGC procedure indeed produces the closure of $C(E)$.

[4] Equivalence of course depends on what relations are considered, but we do not specify this directly to avoid notational clutter.

[5] We note that we could have added an option for symmetric relations, but we do not include these, as they not exhibited in the datasets we consider.

3.2 Calculating Weighted Scene Graph Canonicalization

Thus far we assumed that the sets R_{trans} and R_{conv} were given. Generally, we don't expect this to be the case. We next explain how to construct a model that doesn't have access to these. In this formulation we will add edges with weights, to reflect our level of certainty in adding them. These weights will depend on parameters, which will be learned from data in an end-to-end manner (see Sect. 5). See Fig. 2 for a high level description of the architecture.

Since we don't know which relations are transitive or converses, we assign probabilities to reflect this uncertainty. In the transitive case, for each $r \in \mathcal{R}$ we use a parameter $\theta_r^{trans} \in \mathbb{R}^{|\mathcal{R}|}$ to define the probability that r is transitive:

$$p^{trans}(r) = \sigma(\theta_r^{trans}) \tag{1}$$

where σ is the sigmoid function. For converse relations, we let $p^{conv}(r'|r)$ denote the probability that r' is the converse of r. We add another *empty* relation $r' = \phi$ such that $p^{conv}(\phi|r)$ is the probability that r has no converse in $\mathcal{R}$. This is parameterized via $\theta_{r,r'}^{conv} \in \mathbb{R}^{|\mathcal{R}|\times|\mathcal{R}\cup\phi|}$ which is used to define the distribution:

$$p^{conv}(r'|r) = \frac{e^{\theta_{r,r'}^{conv}}}{\sum_{\hat{r}\in\mathcal{R}\cup\phi} e^{\theta_{r,\hat{r}}^{conv}}} \tag{2}$$

Finally, since converse pairs are typically symmetric (e.g.., "left" is the converse of "right" and vise-versa), for every $r, r' \in \mathcal{R}\times\mathcal{R}$ we set $\theta_{r,r'}^{conv} = \theta_{r',r}^{conv}$. Our model will use these probabilities to complete edges as explained next. In Sect. 3.1 we described the SGC method, which takes a graph E and outputs its completion $C(E)$. The method assumed knowledge of the converse and transitive relations. Here we extend this approach to the case where we have weights on the properties of relations, as per Eq. 1 and 2. Since we have weights on possible completions we will need to work with a weighted relation graph and thus from now on consider edges (i, r, j, w). Below we describe two methods *WSGC-E* and *WSGC-S* for obtaining weighted graphs. Section 4 shows how to use these weighted graphs in an SG to image model.

Exact Weighted Scene Graph Canonicalization (WSGC-E). We describe briefly a method that is a natural extension of SGC (further details are provided in the Supplementary). It begins with the user-specified graph E, with weights of one. Next two weighted completion steps are performed, corresponding to the SGC steps. **Converse Completion:** In SGC, this step adds all converse edges. In the weighted case it makes sense to add the converse edge with its corresponding converse weight. For example, if the graph E contains the edge $(i, \text{above}, j, 1)$ and $p^{conv}(\text{below}|\text{above}) = 0.7$, we add the edge $(j, \text{below}, i, 0.7)$. **Transitive Completion:** In SGC, all transitive edges are found and added. In the weighted case, a natural alternative is to set a weight of a path to be the product of weights along this path, and set the weight of a completed edge (i, r, j) to be the maximum weight of a path between i and j times the probability $p^{trans}(r)$ that the relation is transitive. This can be done in poly-time, but runtime can be substantial for large graphs. We offer a faster approach next.

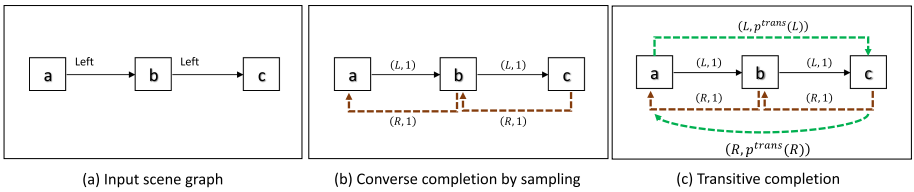


Fig. 3. Illustration of WSGC-S. (a) The input graph. (b) Converse edges (brown arrows) are sampled from p^{conv} and assigned a weight 1 (here two edges were sampled). (c) Transitive edges (green arrows) are completed and assigned a weight p^{trans}.

Sampling Based Weighted Scene Graph Canonicalization (WSGC-S). The difficulty in WSGC-E is that the transitivity step is performed on a dense graph (most weights will be non-zero). To overcome this, we propose to replace the converse completion step of WSGC-E with a sampling based approach that samples completed edges, but always gives them a weight of 1 when they are added. In this way, the transitive step is computed on a much sparser graph with weights 1. We next describe the two steps for the WSGC-S procedure.

Converse Completion: Given the original user-provided graph E, for each r and edge $(i, r, j, 1)$ we sample a random variable $Z \in \mathcal{R} \cup \phi$ from $p^{conv}(\cdot|r)$ and if $Z \neq \phi$, we add the edge $(j, Z, i, 1)$. For example, see Fig. 3b. After sampling such Z for all edges, a new graph E' is obtained, where all the weights are 1.[6]

Transitive Completion: For the graph E' and for each relation r, calculate the transitive closure of $C(E'_r)$ and add all new edges in this closure to E' with weight $p^{trans}(r)$. See illustration in Fig. 3c. Note that this can be calculated in polynomial time using the FW algorithm [10], as in the SGC case.

Finally, we note that if all assigned weights are discrete, both the WSGC-E and WSGC-S are identical to SGC.

4 Scene Graph to Image Using Canonicalization

Thus far we showed how to take the original graph E and complete it into a weighted graph E', using the WSGC-S procedure. Next, we show how to use E' to generate an image, by first mapping E' to a scene layout (see Fig. 2), and then mapping the layout to an image (see AttSPADE Figure in the Supplementary). The following two components are variants of previous SG to image models [1,17,48], and thus we describe them briefly (see Supplementary for details).

From Weighted SG to Layout: A layout is a set of bounding boxes for the nodes in the SG. A natural architecture for such graph-labeling problems is a Graph Convolutional Network (GCN) [22]. Indeed, GCNs have recently been used for the SG to layout task [1,17,25]. We also employ this approach here, but modify it to our weighted scene graph. Namely, we modify the graph convolution

[6] We could sample multiple times and average, but this is not necessary in practice.

layer such that the aggregation step of each node is set to be a weighted average where the weights are those in the canonical SG.

From Layout to Image: We now need to transform the obtained layout in Sect. 4 to an actual image. Several works have proposed models for this step [49,63], where the input was a set of bounding boxes and their object categories. We follow this approach, but extend it so that attributes for each object (e.g., color, shape and material, as in the CLEVR dataset) can be specified. We achieve this via a novel generative model, AttSPADE, that supports attributes. More details are in Supplementary. Figure 4 shows an example of the model trained on CLEVR and applied to several SGs. Finally, our experiments on non CLEVR datasets simply we use a pre-trained LostGAN [48] model.

5 Losses and Training

Thus far we described a model that starts with an SG and outputs an image, using the following three steps: SG to canonical weighted SG (Sect. 3.2), weighted SG to layout (Sect. 4) and finally layout to image (Sect. 4). In this section we describe how the parameters of these steps are trained in an end-to-end manner. We focus on training with the WSGC-S, since this is what we use in most of our experiments. See Supplementary for Training with WSGC-E.

Below we describe the loss for a single input scene graph E and its ground truth layout Y. The parameters of the model are as follows: θ^g are the parameters of the GCN in Sect. 4, θ^{trans} are the parameters of the transitive probability (Eq. 1), and θ^{conv} are those of the converse probability (Eq. 2). Let θ denote the set of all parameters. Recall that in the first step Sect. 3.2, we sample a set of random variables $\bar{Z}$ and use these to obtain a weighted graph $\text{WSGC}_{\bar{Z}}(E;\theta^{trans})$. Denote the GCN applied to this graph by $G_{\theta^g}(\text{WSGC}_{\bar{Z}}(E;\theta^{trans}))$.

We use the L_1 loss between the predicted and ground truth bounding boxes Y. Namely, we wish to minimize the following objective:

$$L(\theta) = \mathbb{E}_{\bar{Z}\backsim q(\theta^{conv})} \left\| Y - G_{\theta^g}(\text{WSGC}_{\bar{Z}}(E;\theta^{trans})) \right\|_1 \tag{3}$$

where $\bar{Z} = \{Z_e | e \in E\}$ is a set of independent random variables each sampled from $p^{conv}(r'|r(e);\theta^{conv})$ (see Eq. 2 and the description of WSGC-E), and $q(\theta^{conv})$ denotes this sampling distribution.

The gradient of this loss with respect to all parameters except θ^{conv} can be easily calculated. Next, we focus on the gradient with respect to θ^{conv}. Because the sampling distribution depends on θ^{conv} it is natural to use the REINFORCE algorithm [54] in this case, as explained next. Define:
$R(\bar{Z};\theta^g,\theta^{trans}) = \|Y - G_{\theta^g}(\text{WSGC}_{\bar{Z}}(E;\theta^{trans}))\|_1$. Then Eq. 3 is: $L(\theta^{conv}) = \mathbb{E}_{\bar{Z}\backsim q(\theta^{conv})} R(\bar{Z};\theta^g,\theta^{trans})$.

The key idea in REINFORCE is the observation that:

$$\nabla_{\theta^{conv}} L(\theta) = \mathbb{E}_{\bar{Z}\backsim q(\theta^{conv})} \nabla_{\theta^{conv}} R(\bar{Z};\theta^g,\theta^{trans}) \log p_\theta^{conv}(\bar{Z})$$

Thus, we can approximate $\nabla_{\theta^{conv}} L(\theta)$ by sampling $\bar{Z}$ and averaging the above.[7]

[7] We sample just one instantiation of $\bar{Z}$ per image, since this works well in practice.

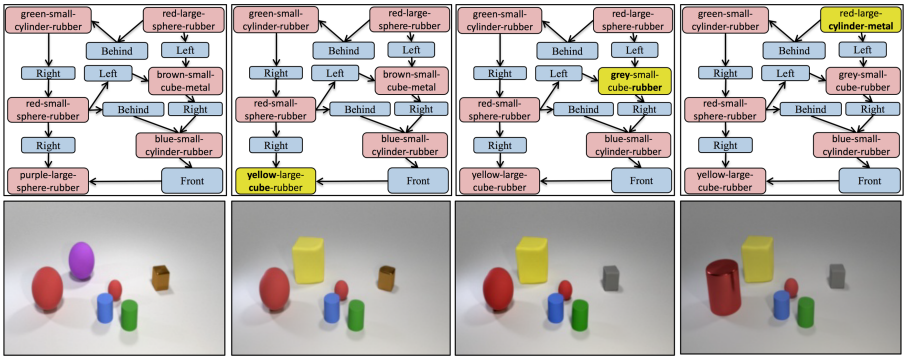


Fig. 4. Demonstration of the AttSPADE generator for scene graphs with varying attributes. Top row shows SGs where each column modifies one attribute. Bottom row is the images generated by AttSPADE.

For the layout-to-image component, most of our experiments use a pre-trained LostGAN model. For CLEVR (Fig. 4) we train our AttSPADE model which is a variant of SPADE [37] and trained similarly (see Supplementary).

6 Experiments

To evaluate our proposed WSGC method, we test performance on two tasks. First, we evaluate on the SG-to-layout task (the task that WSGC is designed for. See Sect. 3.2). We then further use these layouts to generate images and demonstrate that improved layouts also yield improved generated images.

Datasets. We consider the following three datasets: COCO-stuff [4], Visual Genome (VG) [24] and CLEVR [18]. We also created a synthetic dataset to quantify the performance of WSGC in a controlled setting.

Synthetic dataset. To test the contribution of learned transitivity to layout prediction, we generate a synthetic dataset. In this dataset, every object is a square with one of two possible sizes. The set of relations includes: *Above* (transitive), Opposite Horizontally and $XNear$ (non-transitive). To generate training and evaluation data, we uniformly sample coordinates of object centers and object sizes and automatically compute relations among object pairs based on their spatial locations. See Supplementary file for further visual examples.

COCO-Stuff 2017 [4]. Contains pixel-level annotations with 40K train and 5K validation images with bounding boxes and segmentation masks for 80 thing categories, and 91 stuff categories. We use the standard subset proposed in previous works [17], which contains ∼25K training, 1024 validation, and 2048 in test. We use an additional subset we call Packed COCO, containing images with at least 16 objects, resulting in $4,341$ train images, 238 validation, and 238 test.

Visual Genome (VG) [24]. Contains $108,077$ images with SGs. We use the standard subset [17]: 62K training, 5506 validation and 5088 test images. We

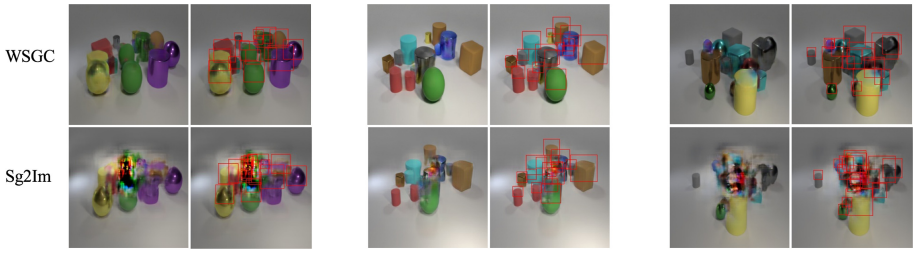


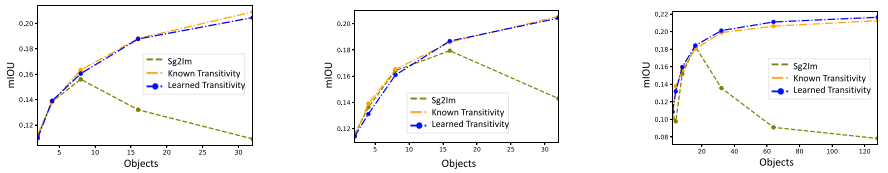
Fig. 5. Examples of image generation for CLEVR where the Sg2Im baseline and our WSGC model were trained on images with a maximum of 10 objects but tested on scenes with 16+ objects. Shown are three examples where: Top row: our WSGC generation (with boxes and without). Bottom row: Sg2Im generation (with boxes and without).

use an additional subset we call Packed VG, containing images with at least 16 objects, resulting in 6341 train images, 809 validation, and 809 test images.
CLEVR [18]. A synthetic dataset based on scene-graphs with four spatial relations: *left*, *right*, *front* and *behind*, as well as attributes *shape*, *size*, *material* and *color*. It has 70k training images and 15k for validation and test.

6.1 Scene-Graph-to-layout Generation

We evaluate the SG-to-layout task using the following metrics: 1) $mIOU$: the mean IOU value. 2) $R@0.3$ and $R@0.5$: the average recall over predictions with IOU greater than 0.3 and 0.5 respectively. We note our WSGC model is identical to the Sg2Im baseline in the SG-to-layout module in all aspects that are not related to canonicalization. This provides a well-controlled ablation showing that canonicalization improves performance.
Testing Robustness to Number of Objects. Scenes can contain a variable number of objects, and SG-to-layout models should work well across these. Here we tested how different models perform as the number of objects is changed in the synthetic dataset. We compared the following models a) A "Learned Transitivity" model that uses WSGC to learn the weights of each relation. b) A "Known Transitivity" model that is given the transitive relations in the data, and performs hard SGC completion (see Sect. 3.1). Comparison between "Learned Transitivity" and "Known Transitivity" is meant to evaluate how well WSGC can learn which relations are transitive. c) A baseline model Sg2Im [17] that does not use any relation completion, but otherwise has the same architecture.

We train these models with two and four GCN layers for up to 32 objects. Additionally, to evaluate generalization to a different number of objects at test time, we train models with eight GCN layers on 16 objects and test on up to 128 objects. Results are shown in Fig. 6a-b. First, it can be seen that the baseline performs significantly worse than transitivity based models. Second, "Learned Transitivity" closely matches "Known Transitivity" indicating that the model



(a) 2 GCN Layers (b) 4 GCN Layers (c) Trained on 16 objects

Fig. 6. Synthetic dataset results. (a-b) The effect of the number of GCN layers on accuracy. Curves denote IOU performance as a function of the number of objects. Each point is a model trained and tested on a fixed number of objects given by the x axis. (c) Out of sample number of objects. The model is trained on 16 objects and evaluated on up to 128 objects.

Table 1. Accuracy of predicted bounding boxes. We consider two different data settings: "Standard" and "Packed". (a) **Standard**: Training and evaluation is on VG images with 3 to 10 objects, and COCO images with 3 to 8 objects. (b) **Packed**: Training and evaluation is on images with 16 or more objects.

Method	Standard						Packed					
	mIOU		R@0.3		R@0.5		mIOU		R@0.3		R@0.5	
	COCO	VG	COCO	VG	COCO	VG	COCO	VG	COCO	VG	COCO	VG
Sg2Im [17] 5 GCN[a]	-	-	52.4	21.9	32.2	10.6	-	-	-	-	-	-
Sg2Im [17] 5 GCN[b]	41.7	16.9	62.6	24.7	37.5	9.7	35.8	25.4	56.0	36.2	25.3	15.8
Sg2Im [17] 8 GCN[b]	41.5	**18.3**	62.9	**26.2**	38.1	10.6	37.2	25.8	58.6	36.9	26.4	15.9
Sg2Im [17] 16 GCN[b]	40.8	16.4	61.4	23.3	36.6	7.8	37.7	27.1	60.3	39.0	26.6	17.0
WSGC 5 GCN (ours)	**41.9**	18.0	**63.3**	25.9	**38.2**	10.6	**39.3**	**28.5**	**62.6**	**42.4**	**30.1**	**18.3**

[a] Results copied from manuscript.
[b] Our implementation of [17]. This is the same as our model without WSGC.

successfully learned which relations are transitive (we also manually confirmed this by inspecting θ^{trans}). Third, the baseline model requires more layers to correctly capture scenes with more objects, whereas our model performs well with two layers. This suggests that WSGC indeed improves generalization ability by capturing invariances. Figure 6c shows that our model also generalizes well when evaluated on a much larger set of objects than what it has seen at training time, whereas the accuracy of the baseline severely degrades in this case.

Layout Accuracy on Packed Scenes. Layout generation is particularly challenging in packed scenes. To quantify this, we evaluate on the Packed COCO and VG datasets. Since Sg2Im [17], PasteGAN [25], and Grid2Im [1] use the same SG-to-layout module, we compare WSGC only to Sg2Im [17]. We test Sg2Im with 5,8 and 16 GCN layers to test the effect of model capacity. The Packed setting in Table 1 shows that WSGC improves layout on all metrics.

We also evaluate on the "standard" COCO/VG setting, which contain relatively few objects, and we therefore do not expect WSGC to improve there. Results in Table 1 show comparable performance to the baselines. In addition, manual inspection revealed that the learned p^{conv} and p^{trans} are overall aligned

Table 2. Evaluating the robustness of the learned canonical representation for models which were trained on Packed COCO. For each SG, a **semantically equivalent** SG is sampled and evaluated at test time. Additionally, models are evaluated on **Noisy SGs**, for which edges contain 10% randomly chosen relations.

Method	Semantically Equivalent			Noisy SGs		
	mIOU	R@0.3	R@0.5	mIOU	R@0.3	R@0.5
Sg2Im [17] 5 GCN^{b}	21.8	29.5	10.7	29.4	42.9	17.8
Sg2Im [17] 8 GCN^{b}	23.6	33.2	11.4	29.9	43.7	18.8
Sg2Im [17] 16 GCN^{b}	21.6	29.0	10.1	28.7	41.8	17.7
WSGC 5 GCN (ours)	**35.3**	**53.2**	**25.7**	**31.8**	**46.6**	**21.9**

[b] Our implementation of [17]. This is the same as our model without WSGC.

Table 3. Results for SG-to-image on **Packed** datasets (16+ objects). For VG and COCO we use the layout-to-image architecture of **LostGAN** [48] and test the effect of different SG-to-layout models. For CLEVR, we use our **AttSPADE** generator.

Method	Inception		Human
	COCO	VG	CLEVR
Sg2Im [17]	5.4 ± 0.3	7.6 ± 1.0	3.2%
WSGC (ours)	$\mathbf{5.6 \pm 0.1}$	$\mathbf{8.0 \pm 1.1}$	96.8%
GT Layout	5.5 ± 0.4	8.2 ± 1.0	-

with expected values (See Supplementary). Finally, the results in the standard setting also show that increasing GCN size for Sg2Im [17] results in overfitting.

Generalization on Semantically Equivalent Graphs. A key advantage of WSGC is that it produces similar layouts for semantically equivalent graphs. This is not true for methods that do not use canonicalization. To test the effectiveness of this property, we modify the test set such that input SGs are replaced with semantically equivalent variations. For example if the original SG was (A, right, B) we may change it to (B, left, A). To achieve this, we generate a semantically equivalent SG by randomly choosing to include or exclude edges which do not change the semantics of the SG. We evaluate on the Packed COCO dataset. Results are shown in Table 2 and qualitative examples are shown in Fig. 7. It can be seen that WSGC significantly outperforms the baselines.

Testing Robustness to Input SGs. Here we ask what happens when input SGs are modified by adding "noisy" edges. This could happen due to noise in the annotation process or even adversarial modifications. Ideally, we would like the generation model to be robust to small SG noise. We next analyze how such modifications affect the model by randomly modifying 10% of the relations in the COCO data. As can be seen in Table 2, the WSGC model can better handle noisy SGs than the baseline. We further note that our model achieves good results on the VG dataset, which was manually annotated, suggesting it is robust to annotation noise. The results in Table 2 also show the Sg2Im generalization

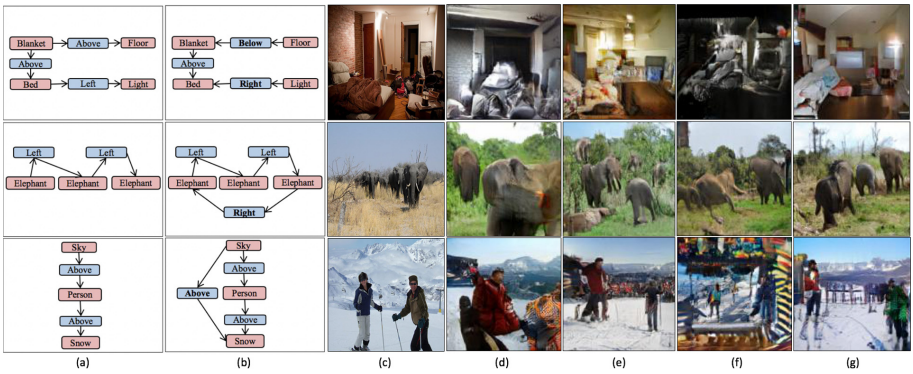


Fig. 7. Generalization from Semantically Equivalent Graphs. Each input SG is changed to a semantically equivalent SG at test time. The layout-to-image model is LostGAN [48] and different SG-to-layout models are tested. (a) Original SG (partial). (b) A modified semantically equivalent SG (partial). (c) GT image. (d-e) Sg2Im [17] and WSGC for the original SG. (f-g) Sg2Im [17] and WSGC for the modified SG.

deteriorates when growing from 8 to 16 layers, suggesting that the effect of canonicalization cannot be achieved by just increasing model complexity.

6.2 Scene-graph-to-image Generation

To test the contribution of our proposed Scene-Graph-to-layout approach to the overall task of SG-to-image generation, we further test it in an end-to-end pipeline for generating images. For Packed COCO and Packed VG, we compare our proposed approach with Sg2Im [17] using a fixed pre-trained LostGAN [49] as the layout-to-image generator. For CLEVR, we use WSGC and our own AttSPADE generator (see Sect. 4). We trained the model on images with a maximum of 10 objects and tested on larger scenes with 16+ objects.

We evaluate performance using Inception score [44] and a study where Amazon Mechanical Turk raters were asked to rank the quality of two images: one generated using our layouts, and the other using SG2Im layouts.[8] Results are provided in Table 3. For COCO and VG it can be seen that WSGC improves the overall quality of generated images. In CLEVR, Table 3, WSGC outperforms Sg2Im in terms of IOU. In 96.8% of the cases, our generated images were ranked higher than SG2Im. Finally, Figs. 5 and 8 provide qualitative examples and comparisons of images generated based on CLEVR and COCO. More generation results on COCO and VG can be seen in the Supplementary.

[8] We used raters only for the CLEVR data, where no GT images or bounding boxes are available for 16+ objects, and thus Inception cannot be evaluated.

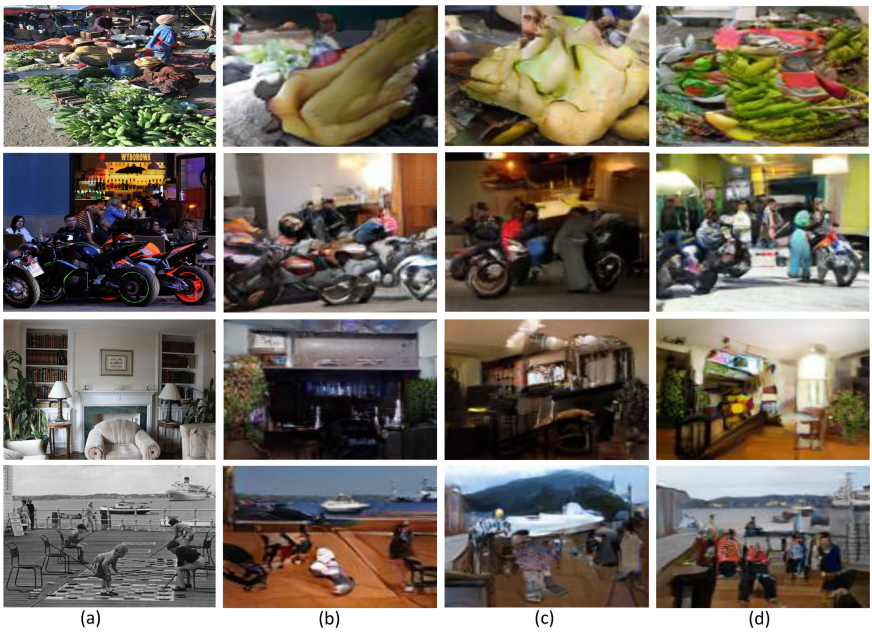


Fig. 8. Selected Scene-graph-to-image generation results on the Packed-COCO dataset. Here, we fix the layout-to-image model to LostGAN [48], while changing different scene graph-to-layout models. (a) GT image. (b) Generation from GT layout. (c) Sg2Im [17] model with LostGAN [48]. (d) Our WSGC model with LostGAN [48].

7 Conclusion

We presented a method for mapping SGs to images that is invariant to a set of logical equivalences. Our experiments show that the method results in improved layouts and image quality. We also observe that canonical representations allow one to handle packed scenes with fewer layers than non-canonical approaches. Intuitively, this is because the closure calculation effectively propagates information across the graph, and thus saves the need for propagation using neural architectures. The advantage is that this step is hard-coded and not learned, thus reducing the size of the model. Our results show the advantage of preprocessing an SG before layout generation. Here we studied this in the context of two types of relation properties. However, it can be extended to more complex ones. In this case, finding the closure will be computationally hard, and would amount to performing inference in Markov Logic Networks [43]. On the other hand, it is likely that modeling such invariances will result in further robustness of the learned models, and is thus an interesting direction for future work.

Acknowledgments. This project has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (grant ERC HOLI 819080). Prof. Darrell's group was supported in part by

DoD, NSF, BAIR, and BDD. This work was completed in partial fulfillment for the Ph.D degree of the first author.

References

1. Ashual, O., Wolf, L.: Specifying object attributes and relations in interactive scene generation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 4561–4569 (2019)
2. Bau, D., et al.: Gan dissection: Visualizing and understanding generative adversarial networks. In: Proceedings of the International Conference on Learning Representations (ICLR) (2019)
3. Brock, A., Donahue, J., Simonyan, K.: Large scale GAN training for high fidelity natural image synthesis. In: International Conference on Learning Representations (2019)
4. Caesar, H., Uijlings, J.R.R., Ferrari, V.: Coco-stuff: Thing and stuff classes in context. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
5. Che, T., Li, Y., Jacob, A.P., Bengio, Y., Li, W.: Mode regularized generative adversarial networks. arXiv preprint arXiv:1612.02136 (2016)
6. Chen, Q., Koltun, V.: Photographic image synthesis with cascaded refinement networks. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1511–1520 (2017)
7. Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., Abbeel, P.: Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In: Advances in neural information processing systems. pp. 2172–2180 (2016)
8. Deng, Z., Chen, J., Fu, Y., Mori, G.: Probabilistic neural programmed networks for scene generation. In: Advances in Neural Information Processing Systems. pp. 4028–4038 (2018)
9. Dong, H., Yu, S., Wu, C., Guo, Y.: Semantic image synthesis via adversarial learning. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 5706–5714 (2017)
10. Floyd, R.W.: Algorithm 97: shortest path. Commun. ACM **5**(6), 345 (1962)
11. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems. pp. 2672–2680 (2014)
12. Herzig, R., et al.: Spatio-temporal action graph networks. In: The IEEE International Conference on Computer Vision (ICCV) Workshops (2019)
13. Herzig, R., Raboh, M., Chechik, G., Berant, J., Globerson, A.: Mapping images to scene graphs with permutation-invariant structured prediction. In: Advances in Neural Information Processing Systems (NIPS) (2018)
14. Hong, S., Yang, D., Choi, J., Lee, H.: Inferring semantic layout for hierarchical text-to-image synthesis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7986–7994 (2018)
15. Huang, X., Liu, M.Y., Belongie, S., Kautz, J.: Multimodal unsupervised image-to-image translation. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 172–189 (2018)
16. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1125–1134 (2017)

17. Johnson, J., Gupta, A., Fei-Fei, L.: Image generation from scene graphs. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
18. Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Zitnick, C.L., Girshick, R.: Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In: CVPR (2017)
19. Johnson, J., Krishna, R., Stark, M., Li, L.J., Shamma, D., Bernstein, M., Fei-Fei, L.: Image retrieval using scene graphs. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3668–3678 (2015)
20. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4401–4410 (2019)
21. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
22. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
23. Krishna, R., Chami, I., Bernstein, M.S., Fei-Fei, L.: Referring relationships. ECCV (2018)
24. Krishna, R., et al.: Visual genome: Connecting language and vision using crowdsourced dense image annotations. ArXiv e-prints (2016)
25. Li, Y., Ma, T., Bai, Y., Duan, N., Wei, S., Wang, X.: Pastegan: A semi-parametric method to generate image from scene graph. In: NeurIPS (2019)
26. Lim, J.H., Ye, J.C.: Geometric gan. arXiv preprint arXiv:1705.02894 (2017)
27. Liu, M.Y., Breuel, T., Kautz, J.: Unsupervised image-to-image translation networks. In: Advances in Neural Information Processing Systems. pp. 700–708 (2017)
28. Mao, X., Li, Q., Xie, H., Lau, Y.R., Wang, Z., Smolley, S.P.: Least squares generative adversarial networks. In: Proceedings of the IEEE International Conference on Computer Vision(2017)
29. Materzynska, J., Xiao, T., Herzig, R., Xu, H., Wang, X., Darrell, T.: Something-else: Compositional action recognition with spatial-temporal interaction networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2020)
30. Mirza, M., Osindero, S.: Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784 (2014)
31. Mittal, G., Agrawal, S., Agarwal, A., Mehta, S., Marwah, T.: Interactive image generation using scene graphs. arXiv preprint arXiv:1905.03743 (2019)
32. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral normalization for generative adversarial networks. In: International Conference on Learning Representations (2018)
33. Nam, S., Kim, Y., Kim, S.J.: Text-adaptive generative adversarial networks: manipulating images with natural language. In: Advances in Neural Information Processing Systems. pp. 42–51 (2018)
34. Odena, A., Olah, C., Shlens, J.: Conditional image synthesis with auxiliary classifier gans. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. pp. 2642–2651. JMLR. org (2017)
35. Van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., et al.: Conditional image generation with pixelcnn decoders. In: Advances in Neural Information Processing Systems. pp. 4790–4798 (2016)
36. Oord, A.v.d., Kalchbrenner, N., Kavukcuoglu, K.: Pixel recurrent neural networks. arXiv preprint arXiv:1601.06759 (2016)

37. Park, T., Liu, M.Y., Wang, T.C., Zhu, J.Y.: Semantic image synthesis with spatially-adaptive normalization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2337–2346 (2019)
38. Qiao, T., Zhang, J., Xu, D., Tao, D.: Mirrorgan: Learning text-to-image generation by redescription. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1505–1514 (2019)
39. Raboh, M., Herzig, R., Chechik, G., Berant, J., Globerson, A.: Differentiable scene graphs. In: Winter Conference on Applications of Computer Vision(2020)
40. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434 (2015)
41. Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., Lee, H.: Generative adversarial text to image synthesis. arXiv preprint arXiv:1605.05396 (2016)
42. Reed, S.E., Akata, Z., Mohan, S., Tenka, S., Schiele, B., Lee, H.: Learning what and where to draw. In: Advances in Neural Information Processing Systems. pp. 217–225 (2016)
43. Richardson, M., Domingos, P.: Markov logic networks. Mach. Learn. **62**(1–2), 107–136 (2006)
44. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training gans. In: Advances in Neural Information Processing Systems. pp. 2234–2242 (2016)
45. Schroeder, B., Tripathi, S., Tang, H.: Triplet-aware scene graph embeddings. In: The IEEE International Conference on Computer Vision (ICCV) Workshops (2019)
46. Schuster, S., Krishna, R., Chang, A., Fei-Fei, L., Manning, C.D.: Generating semantically precise scene graphs from textual descriptions for improved image retrieval. In: Proceedings of The Fourth Workshop on Vision and Language. pp. 70–80 (2015)
47. Sharma, S., Suhubdy, D., Michalski, V., Kahou, S.E., Bengio, Y.: Chatpainter: Improving text to image generation using dialogue. arXiv preprint arXiv:1802.08216 (2018)
48. Sun, W., Wu, T.: Image synthesis from reconfigurable layout and style. In: The IEEE International Conference on Computer Vision (ICCV) (2019)
49. Sun, W., Wu, T.: Image synthesis from reconfigurable layout and style. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 10531–10540 (2019)
50. Taigman, Y., Polyak, A., Wolf, L.: Unsupervised cross-domain image generation. arXiv preprint arXiv:1611.02200 (2016)
51. Tan, F., Feng, S., Ordonez, V.: Text2scene: Generating compositional scenes from textual descriptions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6710–6719 (2019)
52. Tripathi, S., Bhiwandiwalla, A., Bastidas, A., Tang, H.: Heuristics for image generation from scene graphs. In: ICLR LLD Workshop (2019)
53. Wang, T.C., Liu, M.Y., Zhu, J.Y., Tao, A., Kautz, J., Catanzaro, B.: High-resolution image synthesis and semantic manipulation with conditional gans. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(2018)
54. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach. Learn. **8**(3–4), 229–256 (1992)
55. Xiaotian, Q., ZHENG, Q., Ying, C., Rynson, W.: Tell me where i am: Object-level scene context prediction. In: The 32nd meeting of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2019). IEEE (2019)

56. Xu, N., Liu, A.A., Liu, J., Nie, W., Su, Y.: Scene graph captioner: Image captioning based on structural visual representation. J. Vis. Commun. Image Represent. **58**, 477–485 (2019)
57. Xu, T., et al.: Attngan: Fine-grained text to image generation with attentional generative adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1316–1324 (2018)
58. Yin, G., Liu, B., Sheng, L., Yu, N., Wang, X., Shao, J.: Semantics disentangling for text-to-image generation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2327–2336 (2019)
59. Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R.R., Smola, A.J.: Deep sets. In: Advances in Neural Information Processing Systems 30, pp. 3394–3404. Curran Associates, Inc. (2017)
60. Zhang, H., Goodfellow, I., Metaxas, D., Odena, A.: Self-attention generative adversarial networks. In: International Conference Machine Learning (2019)
61. Zhang, H., et al.: Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 5907–5915 (2017)
62. Zhao, B., Meng, L., Yin, W., Sigal, L.: Image generation from layout. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8584–8593 (2019)
63. Zhao, B., Meng, L., Yin, W., Sigal, L.: Image generation from layout. In: CVPR (2019)
64. Zhao, J., Mathieu, M., LeCun, Y.: Energy-based generative adversarial network. arXiv preprint arXiv:1609.03126 (2016)
65. Zhou, X., Huang, S., Li, B., Li, Y., Li, J., Zhang, Z.: Text guided person image synthesis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3663–3672 (2019)
66. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2223–2232 (2017)
67. Zhu, J.Y., et al.: Toward multimodal image-to-image translation. In: Advances in Neural Information Processing Systems. pp. 465–476 (2017)