

Semantic Image Manipulation Using Scene Graphs

Helisa Dhama^{1, *}

Azade Farshad^{1, *}

Iro Laina^{1,2}

Nassir Navab^{1,3}

Gregory D. Hager³

Federico Tombari^{1,4}

Christian Rupprecht²

¹ Technische Universität München ² University of Oxford ³ Johns Hopkins University ⁴ Google

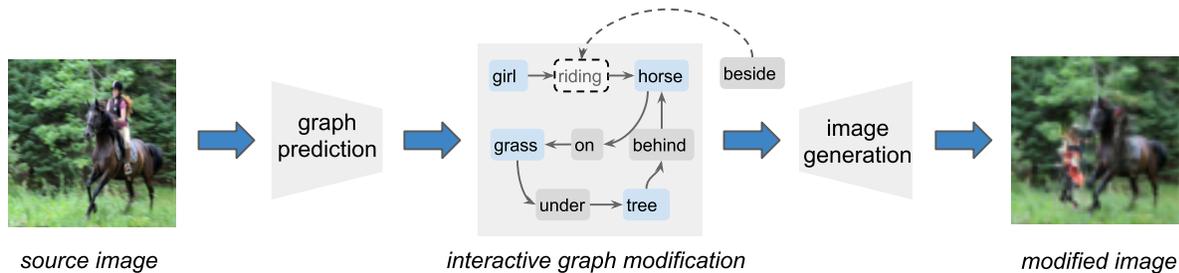


Figure 1: **Semantic Image Manipulation.** Given an image, we predict a semantic scene graph. The user interacts with the graph by making changes on the nodes and edges. Then, we generate a modified version of the source image, which respects the constellations in the modified graph.

Abstract

Image manipulation can be considered a special case of image generation where the image to be produced is a modification of an existing image. Image generation and manipulation have been, for the most part, tasks that operate on raw pixels. However, the remarkable progress in learning rich image and object representations has opened the way for tasks such as text-to-image or layout-to-image generation that are mainly driven by semantics. In our work, we address the novel problem of image manipulation from scene graphs, in which a user can edit images by merely applying changes in the nodes or edges of a semantic graph that is generated from the image. Our goal is to encode image information in a given constellation and from there on generate new constellations, such as replacing objects or even changing relationships between objects, while respecting the semantics and style from the original image. We introduce a spatio-semantic scene graph network that does not require direct supervision for constellation changes or image edits. This makes it possible to train the system from existing real-world datasets with no additional annotation effort.

*The first two authors contributed equally to this work
Project page: <https://he-dhama.github.io/SIMSG/>

1. Introduction

The goal of image understanding is to extract rich and meaningful information from an image. Recent techniques based on deep representations are continuously pushing the boundaries of performance in recognizing objects [39] and their relationships [29] or producing image descriptions [19]. Understanding is also necessary for image synthesis, e.g. to generate natural looking images from an abstract semantic canvas [4, 47, 60] or even from language descriptions [11, 26, 38, 56, 58]. High-level image manipulation, however, has received less attention. Image manipulation is still typically done at pixel level via photo editing software and low-level tools such as in-painting. Instances of higher-level manipulation are usually object-centric, such as facial modifications or reenactment. A more abstract way of manipulating an image from its semantics, which includes objects, their relationships and attributes, could make image editing easier with less manual effort from the user.

In this work, we present a method to perform semantic editing of an image by modifying a scene graph, which is a representation of the objects, attributes and interactions in the image (Figure 1). As we show later, this formulation allows the user to choose among different editing functions. For example, instead of manually segmenting, deleting and in-painting unwanted tourists in a holiday photo, the user

can directly manipulate the scene graph and delete selected `<person>` nodes. Similarly, graph nodes can be easily replaced with different semantic categories, for example replacing `<clouds>` with `<sky>`. It is also possible to re-arrange the spatial composition of the image by swapping people or object nodes on the image canvas. To the best of our knowledge, this is the first approach to image editing that also enables semantic relationship changes, for example changing “*a person walking in front of the sunset*” to “*a person jogging in front of the sunset*” to create a more scenic image. The capability to reason and manipulate a scene graph is not only useful for photo editing. The field of robotics can also benefit from this kind of task, *e.g.* a robot tasked to tidy up a room can — prior to acting — manipulate the scene graph of the perceived scene by moving objects to their designated spaces, changing their relationships and attributes: “*clothes lying on the floor*” to “*folded clothes on a shelf*”, to obtain a realistic future view of the room.

Much previous work has focused either on generating a scene graph from an image [27, 31] or an image from a graph [1, 17]. Here we face challenges unique to the combined problem. For example, if the user changes a relationship attribute — *e.g.* `<boy, sitting on, grass>` to `<boy, standing on, grass>`, the system needs to generate an image that contains the *same* boy, thus preserving the identity as well as the content of the rest of the scene. Collecting a fully supervised data set, *i.e.* a data set of “before” and “after” pairs together with the associated scene graph, poses major challenges. As we discuss below, this is not necessary. It is in fact possible to learn how to modify images using only training pairs of images and scenes graphs, which is data already available.

In summary, we present a novel task; given an image, we manipulate it using the respective scene graph. Our contribution is a method to address this problem that does not require full supervision, *i.e.* image pairs that contain scene changes. Our approach can be seen as semi-automatic, since the user does not need to manually edit the image but indirectly interacts with it through the nodes and edges of the graph. In this way, it is possible to make modifications with respect to visual entities in the image and the way they interact with each other, both spatially and semantically. Most prominently, we achieve various types of edits with a single model, including semantic relationship changes between objects. The resulting image preserves the original content, but allows the user to flexibly change and/or integrate new or modified content as desired.

2. Related Work

Conditional image generation The success of deep generative models [8, 22, 37, 45, 46] has significantly contributed to advances in (un)conditional image synthesis. Conditional image generation methods model the conditional

distribution of images given some prior information. For example, several practical tasks such as denoising or inpainting can be seen as generation from noisy or partial input. Conditional models have been studied in literature for a variety of use cases, conditioning the generation process on image labels [30, 32], attributes [49], lower resolution images [25], semantic segmentation maps [4, 47], natural language descriptions [26, 38, 56, 58] or generally translating from one image domain to another using paired [14] or unpaired data [63]. Most relevant to our approach are methods that generate natural scenes from layout [11, 60] or scene graphs [17].

Image manipulation Unconditional image synthesis is still an open challenge when it comes to complex scenes. Image manipulation, on the other hand, focuses on image parts in a more constrained way that allows to generate better quality samples. Image manipulation based on semantics has been mostly restricted to object-centric scenarios; for example, editing faces automatically using attributes [5, 24, 59] or via manual edits with a paintbrush and scribbles [3, 62]. Also related is image composition which also makes use of individual objects [2] and faces the challenge of decoupling appearance and geometry [55].

On the level of scenes, the most common examples based on generative models are inpainting [35], in particular conditioned on semantics [52] or user-specified contents [16, 61], as well as object removal [7, 42]. Image generation from semantics also supports interactive editing by applying changes to the semantic map [47]. Differently, we follow a semi-automatic approach to address all these scenarios using a single general-purpose model and incorporating edits by means of a scene graph. On another line, Hu *et al.* [12] propose a hand-crafted image editing approach, which uses graphs to carry out library-driven replacement of image patches. While [12] focuses on copy-paste tasks, our framework allows for high-level semantic edits and deals with object deformations.

Our method is trained by reconstructing the input image so it does not require paired data. A similar idea is explored by Yao *et al.* [51] for 3D-aware modification of a scene (*i.e.* 3D object pose) by disentangling semantics and geometry. However, this approach is limited to a specific type of scenes (streets) and target objects (cars) and requires CAD models. Instead, our approach addresses semantic changes of objects and their relationships in natural scenes, which is made possible using scene graphs.

Images and scene graphs Scene graphs provide abstract, structured representations of image content. Johnson *et al.* [20] first defined a scene graph as a directed graph representation that contains objects and their attributes and relationships, *i.e.* how they interact with each other. Fol-

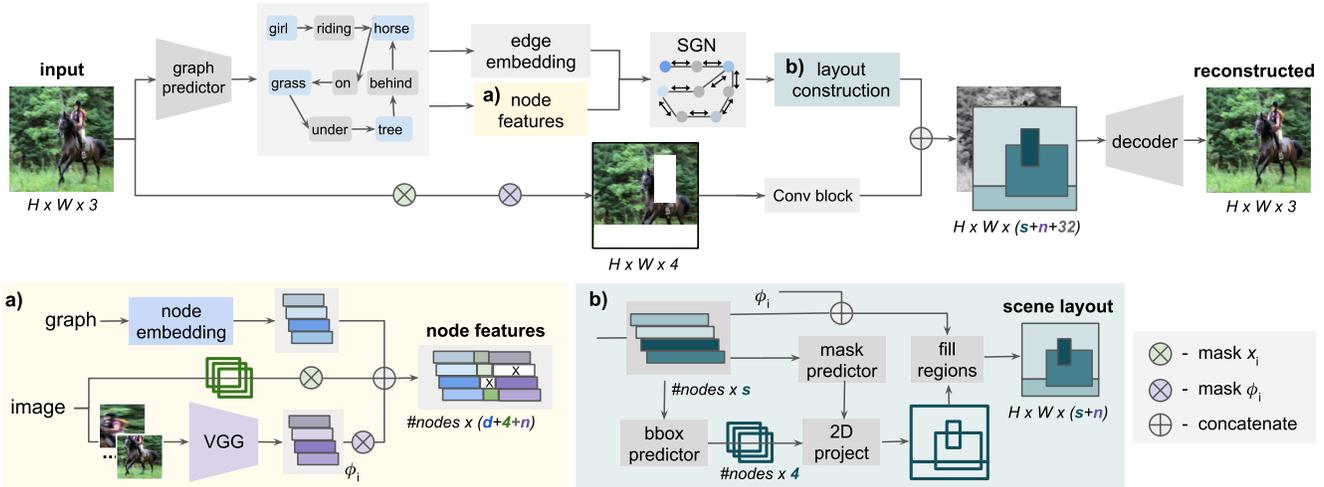


Figure 2: **Overview of the training strategy.** *Top:* Given an image, we predict its scene graph and reconstruct the input from a masked representation. a) The graph nodes o_i (blue) are enriched with bounding boxes x_i (green) and visual features ϕ_i (violet) from cropped objects. We randomly mask boxes x_i , object visual features ϕ_i and the source image; the model then reconstructs the same graph and image utilizing the remaining information. b) The per-node feature vectors are projected to 2D space, using the bounding box predictions from SGN.

lowing this graph representation paradigm, different methods have been proposed to generate scene graphs from images [9, 27, 28, 31, 36, 48, 50, 54]. By definition, scene graph generation mainly relies on successfully detecting visual entities in the image (object detection) [39] and recognizing how these entities interact with each other (visual relationship detection) [6, 15, 29, 40, 53].

The reverse and under-constrained problem is to generate an image from its scene graph, which has been recently addressed by Johnson *et al.* using a graph convolution network (GCN) to decode the graph into a layout and consecutively translate it into image [17]. We build on this architecture and propose additional mechanisms for information transfer from an image that act as conditioning for the system, when the goal is image editing and not free-form generation. Also related is image generation directly from layouts [60]. Very recent related work focuses on interactive image generation from scene graphs [1] or layout [44]. These methods differ from ours in two aspects. First, while [1, 44] process a graph/layout to generate multiple variants of an image, our method manipulates an *existing* image. Second, we present complex semantic relationship editing, while they use graphs with simplified spatial relations — *e.g.* relative object positions such as `left of` or `above` in [1] — or without relations at all, as is the case for the layout-only approach in [44].

3. Method

The focus of this work is to perform semantic manipulation of images without direct supervision for image edits,

i.e. without paired data of original and modified content. Starting from an input image I , we generate its scene graph \mathcal{G} that serves as the means of interaction with a user. We then generate a new image I' from the user-modified graph representation $\hat{\mathcal{G}}$ and the original content of I . An overview of the method is shown in Figure 1. Our method can be split into three interconnected parts. The first step is scene graph generation, where we encode the image contents in a spatio-semantic scene graph, designed so that it can easily be manipulated by a user. Second, during inference, the user manipulates the scene graph by modifying object categories, locations or relations by directly acting on the nodes and edges of the graph. Finally, the output image is generated from the modified graph. Figure 2 shows the three components and how they are connected.

A particular challenge in this problem is the difficulty in obtaining training data, *i.e.* matching pairs of source and target images together with their corresponding scene graphs. To overcome these limitations, we demonstrate a method that learns the task by image reconstruction in an unsupervised way. Due to readily available training data, graph prediction instead is learned with full supervision.

3.1. Graph Generation

Generating a scene graph from an image is a well-researched problem [27, 31, 48, 54] and amounts to describing the image with a directed graph $\mathcal{G} = (\mathcal{O}, \mathcal{R})$ of objects \mathcal{O} (nodes) and their relations \mathcal{R} (edges). We use a state-of-the-art method for scene graph prediction (F-Net) [27] and build on its output. Since the output of the system is

a generated image, our goal is to encode as much image information in the scene graph as possible — additional to semantic relationships. We thus define objects as triplets $o_i = (c_i, \phi_i, x_i) \in \mathcal{O}$, where $c_i \in \mathbb{R}^d$ is a d -dimensional, learned embedding of the i -th object category and $x_i \in \mathbb{R}^4$ represents the four values defining the object’s bounding box. $\phi_i \in \mathbb{R}^n$ is a visual feature encoding of the object which can be obtained from a convolutional neural network (CNN) pre-trained for image classification. Analogously, for a given relationship between two objects i and j , we learn an embedding ρ_{ij} of the relation class $r_{ij} \in \mathcal{R}$.

One can also see this graph representation as an augmentation of a simple graph—that only contains object and predicate categories—with image features and spatial locations. Our graph contains sufficient information to preserve the identity and appearance of objects even when the corresponding locations and/or relationships are modified.

3.2. Spatio-semantic Scene Graph Network

At the heart of our method lies the spatio-semantic scene graph network (SGN) that operates on the (user-) modified graph. The network learns a graph transformation that allows information to flow between objects, along their relationships. The task of the SGN is to learn robust object representations that will be then used to reconstruct the image. This is done by a series of convolutional operations on the graph structure.

The graph convolutions are implemented by an operation τ_e on edges of the graph

$$(\alpha_{ij}^{(t+1)}, \rho_{ij}^{(t+1)}, \beta_{ij}^{(t+1)}) = \tau_e \left(\nu_i^{(t)}, \rho_{ij}^{(t)}, \nu_j^{(t)} \right), \quad (1)$$

with $\nu_i^{(0)} = o_i$, where t represents the layer of the SGN and τ_e is implemented as a multi-layer perceptron (MLP). Since nodes can appear in several edges, the new node feature $\nu_i^{(t+1)}$ is computed by averaging the results from the edge-wise transformation, followed by another projection τ_n

$$\nu_i^{(t+1)} = \tau_n \left(\frac{1}{N_i} \left(\sum_{j|(i,j) \in \mathcal{R}} \alpha_{ij}^{(t+1)} + \sum_{k|(k,i) \in \mathcal{R}} \beta_{ki}^{(t+1)} \right) \right) \quad (2)$$

where N_i represents the number of edges that start or end in node i . After T graph convolutional layers, the last layer predicts one latent representation per node, *i.e.* per object. This output object representation consists of predicted bounding box coordinates $\hat{x}_i \in \mathbb{R}^4$, a spatial binary mask $\hat{m}_i \in \mathbb{R}^{M \times M}$ and a node feature vector $\psi_i \in \mathbb{R}^s$. Predicting coordinates for each object is a form of reconstruction, since object locations are known and are already encoded in the input o_i . As we show later, this is needed when modifying the graph, for example for a new node to be added. The predicted object representation will be then reassembled into the spatial configuration of an image, as the scene layout.

3.3. Scene Layout

The next component is responsible for transforming the graph-structured representations predicted by the SGN back into a 2D spatial arrangement of features, which can then be decoded into an image. To this end, we use the predicted bounding box coordinates \hat{x}_i to project the masks \hat{m}_i in the proper region of a 2D representation of the same resolution as the input image. We concatenate the original visual feature ϕ_i with the node features ψ_i to obtain a final node feature. The projected mask region is then filled with the respective features, while the remaining area is padded with zeros. This process is repeated for all objects, resulting in $|\mathcal{O}|$ tensors of dimensions $(n + s) \times H \times W$, which are aggregated through summation into a single layout for the image. The output of this component is an intermediate representation of the scene, which is rich enough to reconstruct an image.

3.4. Image Synthesis

The last part of the pipeline is the task of synthesizing a target image from the information in the source image I and the layout prediction. For this task, we employ two different decoder architectures, cascaded refinement networks (CRN) [4] (similar to [17]), as well as SPADE [34], originally proposed for image synthesis from a semantic segmentation map. We condition the image synthesis on the source image by concatenating the predicted layout with extracted low-level features from the source image. In practice, prior to feature extraction, regions of I are occluded using a mechanism explained in Section 3.5. We fill these regions with Gaussian noise to introduce stochasticity for the generator.

3.5. Training

Training the model with full supervision would require annotations in the form of quadruplets $(I, \mathcal{G}, \mathcal{G}', I')$ where an image I is annotated with a scene graph \mathcal{G} , a modified graph \mathcal{G}' and the resulting modified image I' . Since acquiring ground truth (I', \mathcal{G}') is difficult, our goal is to train a model supervised only by (I, \mathcal{G}) through reconstruction. Thus, we generate annotation quadruplets $(\tilde{I}, \tilde{\mathcal{G}}, \mathcal{G}, I)$ using the available data (I, \mathcal{G}) as the *target* supervision and simulate $(\tilde{I}, \tilde{\mathcal{G}})$ via a random masking procedure that operates on object instances. During training, an object’s visual features ϕ_i are masked with probability p_ϕ . Independently, we mask the bounding box x_i with probability p_x . When “hiding” input information, image regions corresponding to the hidden nodes are also occluded prior to feature extraction.

Effectively, this masking mechanism transforms the editing task into a reconstruction problem. At run time, a real user can directly edit the nodes or edges of the scene graph. Given the edit, the image regions subject to modification are occluded, and the network, having learned to reconstruct the image from the scene graph, will create a plausible modified image. Consider the example of a person riding a horse

(Figure 1). The user wishes to apply a change in the way the two entities interact, modifying the predicate from `riding` to `beside`. Since we expect the spatial arrangement to change, we also discard the localization x_i of these entities in the original image; their new positions \hat{x}_i will be estimated given the layout of the rest of the scene (e.g. grass, trees). To encourage this change, the system should automatically mask the original image regions related to the target objects. However, to ensure that the *visual identities* of horse and rider are preserved through the change, their visual feature encodings ϕ_i must remain unchanged.

We use a combination of loss terms to train the model. The bounding box prediction is trained by minimizing the L_1 -norm: $\mathcal{L}_b = \|x_i - \hat{x}_i\|_1$, with weighting term λ_b . The image generation task is learned by adversarial training with two discriminators. A local discriminator D_{obj} operates on each reconstructed region to ensure that the generated patches look realistic. We also apply an auxiliary classifier loss [33] to ensure that D_{obj} is able to classify the generated objects into their real labels. A global discriminator D_{global} encourages consistency over the entire image. Finally, we apply a photometric loss term $\mathcal{L}_r = \|I - I'\|_1$ to enforce the image content to stay the same in regions that are not subject to change. The total synthesis loss is then

$$\begin{aligned} \mathcal{L}_{\text{synthesis}} = & \mathcal{L}_r + \lambda_g \min_G \max_D \mathcal{L}_{\text{GAN,global}} \\ & + \lambda_o \min_G \max_D \mathcal{L}_{\text{GAN,obj}} + \lambda_a \mathcal{L}_{\text{aux,obj}}, \end{aligned} \quad (3)$$

where $\lambda_g, \lambda_o, \lambda_a$ are weighting factors and

$$\mathcal{L}_{\text{GAN}} = \mathbb{E}_{q \sim p_{\text{real}}} \log D(q) + \mathbb{E}_{q \sim p_{\text{fake}}} \log(1 - D(q)), \quad (4)$$

where p_{real} corresponds to the ground truth distribution (of each object or the whole image) and p_{fake} is the distribution of generated (edited) images or objects, while q is the input to the discriminator which is sampled from the real or fake distributions. When using SPADE, we additionally employ a perceptual loss term $\lambda_p \mathcal{L}_p$ and a GAN feature loss term $\lambda_f \mathcal{L}_f$ following the original implementation [34]. Moreover, D_{global} becomes a multi-scale discriminator.

Full implementation details regarding the architectures, hyper-parameters and training can be found in the Appendix.

4. Experiments

We evaluate our method quantitatively and qualitatively on two datasets, CLEVR [18] and Visual Genome [23], with two different motivations. As CLEVR is a synthetic dataset, obtaining ground truth pairs for image editing is possible, which allows quantitative evaluation of our method. On the other hand, experiments on Visual Genome (VG) show the performance of our method in a real, much less constrained, scenario. In absence of source-target image pairs in VG, we

Method	All pixels				RoI only	
	MAE ↓	SSIM ↑	LPIPS ↓	FID ↓	MAE ↓	SSIM ↑
Full-sup	6.75	97.07	0.035	3.35	9.34	93.49
Ours (CRN)	7.83	96.16	0.036	6.32	10.09	93.54
Ours (SPADE)	5.47	96.51	0.035	4.73	7.22	94.98

Table 1: **Image manipulation on CLEVR.** We compare our method with a fully-supervised baseline. Detailed results for all modification types are reported in the Appendix.

evaluate an image in-painting proxy task and compare to a baseline based on `sg2im` [17]. We report results for standard image reconstruction metrics: the structural similarity index (SSIM), mean absolute error (MAE) and perceptual error (LPIPS) [57]. To assess the image generation quality and diversity, we report the commonly used inception score (IS) [41] and the FID [10] metric.

Conditional `sg2im` baseline (Cond-`sg2im`). We modify the model of [17] to serve as a baseline. Since their method generates images directly from scene graphs without a source image, we condition their image synthesis network on the input image by concatenating it with the layout component (instead of noise in the original work). To be comparable to our approach, we mask image regions corresponding to the target objects prior to concatenation.

Modification types. Since image editing using scene graphs is a novel task, we define several modification modes, depending on how the user interacts with the graph. **Object removal:** A node is removed entirely from the graph together with all the edges that connect this object with others. The source image region corresponding to the object is occluded. **Object replacement:** A node is assigned to a different semantic category. We do not remove the full node; however, the visual encoding ϕ_i of the original object is set to zero, as it does not describe the novel object. The location of the original entity is used to keep the new object in place, while size comes from the bounding box estimated from the SGN, to fit the new category. **Relationship change:** This operation usually involves re-positioning of entities. The goal is to keep the subject and object but change their interaction, e.g. `<sitting>` to `<standing>`. Both the original and novel appearance image regions are occluded, to enable background in-painting and target object generation. The visual encodings ϕ_i are used to condition the SGN and maintain the visual identities of objects on re-appearance.

4.1. Synthetic Data

We use the CLEVR framework [18] to generate a dataset (for details please see the Appendix) of image and scene

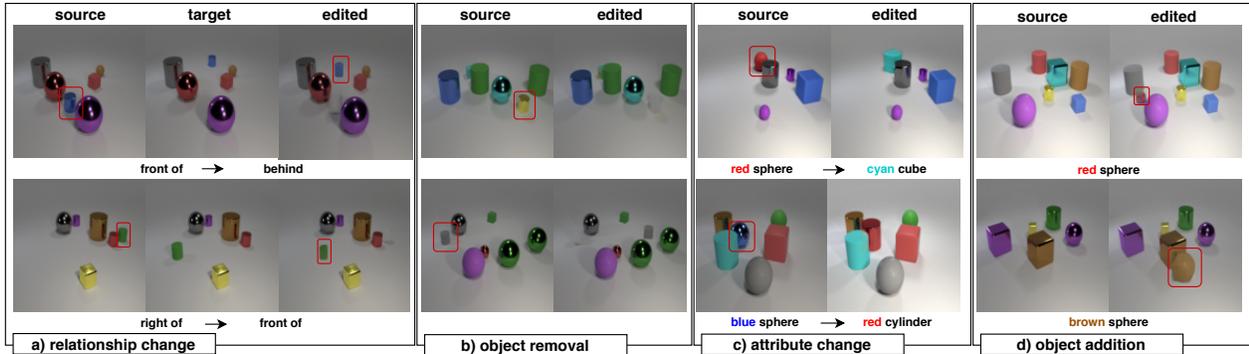


Figure 3: **Image manipulation on CLEVR** We compare different changes in the scene including changing the relationship between two objects, node removal and changing a node (corresponding to attribute changing).

graph editing pairs $(I, \mathcal{G}, \mathcal{G}', I')$, to evaluate our method with exact ground truth.

We train our model *without making use of image pairs* and compare our approach to a fully-supervised setting. When training with full supervision the complete source image and target graph are given to the model and the model is trained by minimizing the \mathcal{L}_1 loss to the ground truth target image instead of the proposed masking scheme.

Table 1 reports the mean SSIM, MAE, LPIPS and FID on CLEVR for the manipulation task (replacement, removal, relationship change and addition). Our method performs better or on par with the fully-supervised setting, on the reconstruction metrics, which shows the capability of synthesizing meaningful changes. The FID results suggest that additional supervision for pairs, if available, would lead to improvement in the visual quality. Figure 3 shows qualitative results of our model on CLEVR. At test time, we apply changes to the scene graph in four different modes: changing relationships (a), removing an object (b), adding an object (d) or changing its identity (c). We highlight the modification with a bounding box drawn around the selected object.

4.2. Real Images

We evaluate our method on Visual Genome [23] to show its performance on natural images. Since there is no ground truth for modifications, we formulate the quantitative evaluation as image reconstruction. In this case, objects are occluded from the original image and we measure the quality of the reconstruction. The qualitative results better illustrate the full potential of our method.

Feature encoding. First, we quantify the role of the visual feature ϕ_i in encoding visual appearance. For a given image and its graph, we use all the associated object locations x_i and visual features (w/ ϕ_i) to condition the SGN. However, the region of the conditioning image corresponding to a candidate node is masked. The task can be interpreted as

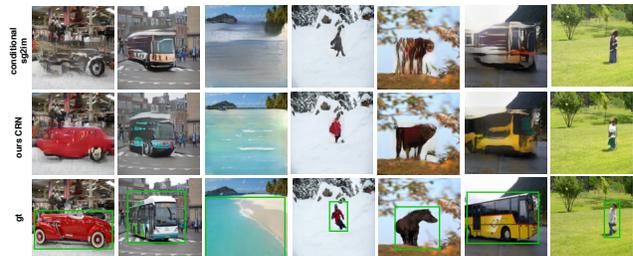


Figure 4: **Visual feature encoding.** Comparison between the baseline (top) and our method (center). The scene graph remains unchanged; an object in the image is occluded, while ϕ_i and x_i are active. Our latent features ϕ_i preserve appearance when the objects are masked from the image.

conditional in-painting. We test our approach in two scenarios; using ground truth graphs (GT) and graphs predicted from the input images (P). We evaluate over all objects in the test set and report the results in Table 2, measuring the reconstruction error a) over all pixels and b) in the target area only (RoI). We compare to the same model without using visual features (w/o ϕ_i) but only the object category to condition the SGN. Naturally, in all cases, including the missing region’s visual features improves the reconstruction metrics (MAE, SSIM, LPIPS). In contrast, inception score and FID remain similar, as these metrics do not consider similarity between direct corresponding pairs of generated and ground truth images. From Table 2 one can observe that while both decoders perform similarly in reconstruction metrics (CRN is slightly better), SPADE dominates for the FID and inception score, indicating higher visual quality.

To evaluate our method in a fully generative setting, we mask the whole image and only use the encoded features ϕ_i for each object. We compare against the state of the art in interactive scene generation (ISG) [1], evaluated in the same setting. Since our main focus is on semantically rich relations, we trained [1] on Visual Genome, utilizing their

Method	Decoder	All pixels					RoI only	
		MAE ↓	SSIM ↑	LPIPS ↓	FID ↓	IS ↑	MAE ↓	SSIM ↑
ISG [1] (Generative, GT)	Pix2pixHD	46.44	28.10	0.32	58.73	6.64±0.07	-	-
Ours (Generative, GT)	CRN	41.57	33.9	0.34	89.55	6.03±0.17	-	-
Ours (Generative, GT)	SPADE	41.88	34.89	0.27	44.27	7.86±0.49	-	-
Cond-sg2im [17] (GT)	CRN	14.25	84.42	0.081	13.40	11.14±0.80	29.05	52.51
Ours (GT) w/o ϕ_i	CRN	9.83	86.52	0.073	10.62	11.45±0.61	27.16	52.01
Ours (GT) w/ ϕ_i	CRN	7.43	88.29	0.058	11.03	11.22±0.52	20.37	60.03
Ours (GT) w/o ϕ_i	SPADE	10.36	86.67	0.069	8.09	12.05±0.80	27.10	54.38
Ours (GT) w/ ϕ_i	SPADE	8.53	87.57	0.051	7.54	12.07±0.97	21.56	58.60
Ours (P) w/o ϕ_i	CRN	9.24	87.01	0.075	18.09	10.67±0.43	29.08	48.62
Ours (P) w/ ϕ_i	CRN	7.62	88.31	0.063	19.49	10.18±0.27	22.89	55.07
Ours (P) w/o ϕ_i	SPADE	13.16	84.61	0.083	16.12	10.45±0.15	32.24	47.25
Ours (P) w/ ϕ_i	SPADE	13.82	83.98	0.077	16.69	10.61±0.37	28.82	49.34

Table 2: **Image reconstruction on Visual Genome.** We report the results using ground truth scene graphs (GT) and predicted scene graphs (P). (Generative) indicates experiments in full generative setting, *i.e.* the whole input image is masked out.

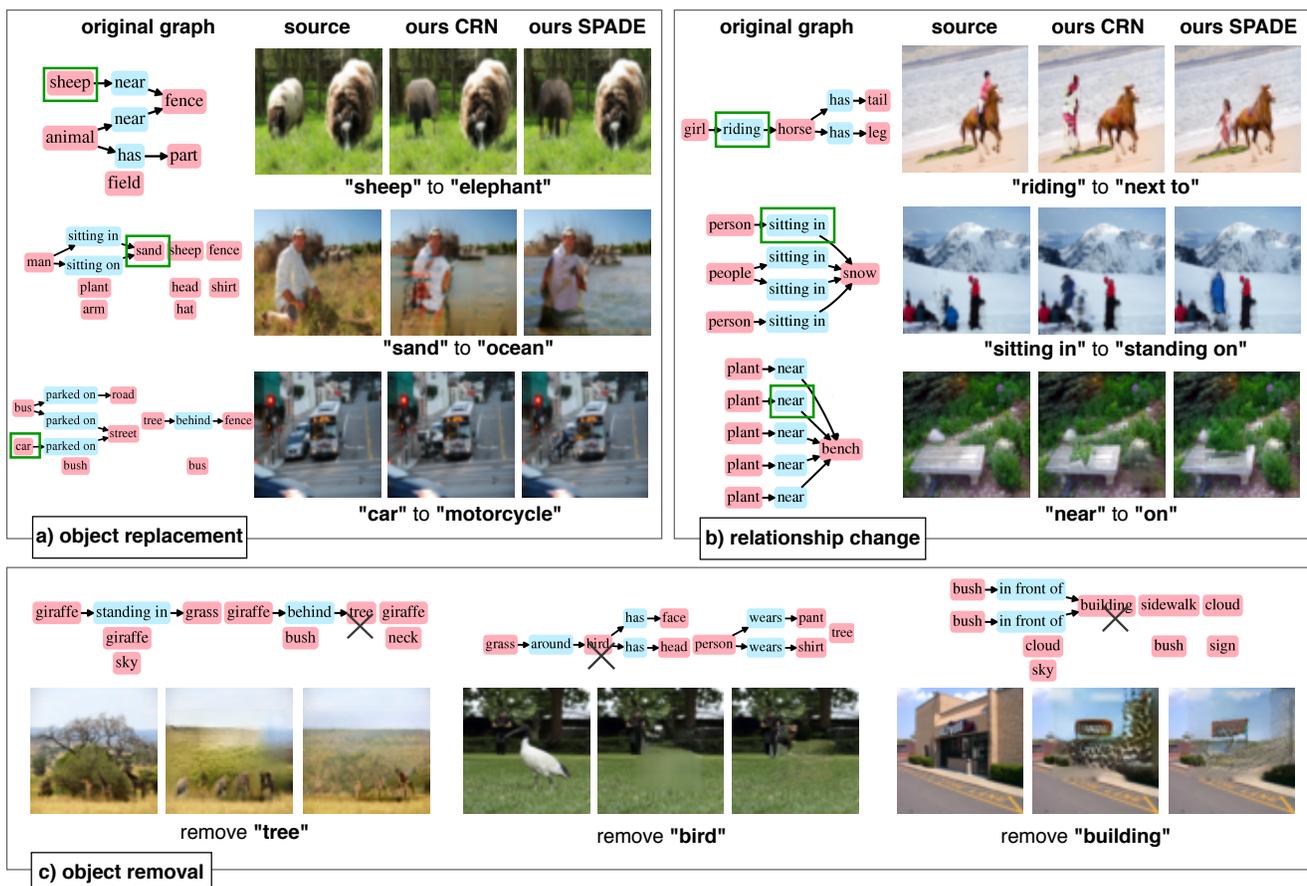


Figure 5: **Image manipulation** Given the source image and the GT scene graph, we semantically edit the image by changing the graph. **a)** object replacement, **b)** relationship changes, **c)** object removal. Green box indicates the changed node or edge.

publicly available code. Table 2 shows comparable reconstruction errors for the generative task, while we clearly

outperform [1] when a source image is given. This motivates our choice of directly manipulating an existing image, rather

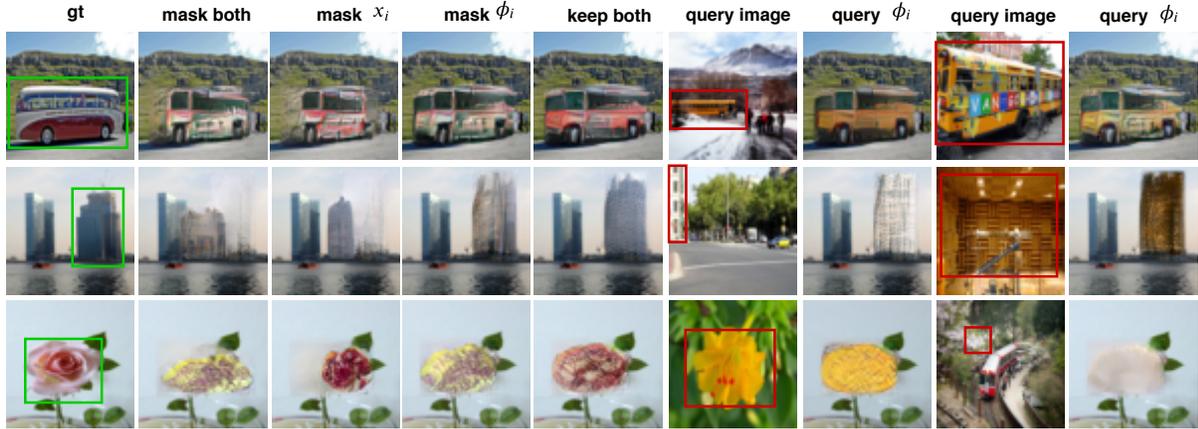


Figure 6: **Ablation of the method components** We present all the different combinations in which the method operates - *i.e.* masked vs. active bounding boxes x_i and/or visual features ϕ_i . When using a query image, we extract visual features of the object annotated with a red bounding box and update the node of an object of the same category in the original image.

than fusing different node features, as parts of the image need to be preserved. Inception score and FID mostly depend on the decoder architecture, where SPADE outperforms Pix2pixHD and CRN.

Figure 4 illustrates qualitative examples. It can be seen that both our method and the cond-sg2im baseline, generate plausible object categories and shapes. However, with our approach, visual features from the original image can be successfully transferred to the output. In practice, this property is particularly useful when we want to re-position objects in the image without changing their identity.

Main task: image editing. We illustrate visual results in three different settings in Figure 5 — object removal, replacement and relationship changes. All image modifications are made by the user at test time, by changing nodes or edges in the graph. We show diverse replacements (a), from small objects to background components. The novel entity adapts to the image context, *e.g.* the ocean (second row) does not occlude the person, which we would expect in standard inpainting. A more challenging scenario is to change the way two objects interact, which typically involves re-positioning. Figure 5 (b) shows that the model can differentiate between semantic concepts, such as *sitting vs. standing* and *riding vs. next to*. The objects are rearranged meaningfully according to the change in relationship type. In the case of object removal (c), the method performs well for backgrounds with uniform texture, but can also handle more complex structures, such as the background in the first example. Interestingly, when the building on the rightmost example is removed, the remaining sign is improvised standing in the bush. More results are shown in the Appendix.

Component ablation. In Figure 6 we qualitatively ablate the components of our method. For a certain image, we mask out a certain object instance which we aim to reconstruct. We test the method under all the possible combinations of masking bounding boxes x_i and/or visual features ϕ_i from the augmented graph representation. Since it might be of interest to in-paint the region with a different object (changing either the category or style), we also experiment with an additional setting, in which external visual features ϕ are extracted from an image of the query object. Intuitively, masking the box properties leads to a small shift in the location and size of the reconstructed object, while masking the object features can result in an object with a different identity than that in the original image.

5. Conclusion

We have presented a novel task — semantic image manipulation using scene graphs — and have shown a novel approach to tackle the learning problem in a way that does not require training pairs of original and modified image content. The resulting system provides a way to change both the content and relationships among scene entities by directly interacting with the nodes and edges of the scene graph. We have shown that the resulting system is competitive with baselines built from existing image synthesis methods, and qualitatively provides compelling evidence for its ability to support modification of real-world images. Future work will be devoted to further enhancing these results, and applying them to both interactive editing and robotics applications.

Acknowledgements We gratefully acknowledge the Deutsche Forschungsgemeinschaft (DFG) for supporting this research work, under the project #381855581. Christian Rupprecht is supported by ERC IDIU-638009.

References

- [1] Oron Ashual and Lior Wolf. Specifying object attributes and relations in interactive scene generation. In *ICCV*, pages 4561–4569, 2019.
- [2] Samaneh Azadi, Deepak Pathak, Sayna Ebrahimi, and Trevor Darrell. Compositional gan: Learning conditional image composition. *arXiv preprint arXiv:1807.07560*, 2018.
- [3] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Neural photo editing with introspective adversarial networks. *arXiv preprint arXiv:1609.07093*, 2016.
- [4] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *ICCV*, pages 1511–1520, 2017.
- [5] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *CVPR*, pages 8789–8797, 2018.
- [6] Bo Dai, Yuqi Zhang, and Dahua Lin. Detecting visual relationships with deep relational networks. In *CVPR*, pages 3076–3086, 2017.
- [7] Helisa Dhama, Nassir Navab, and Federico Tombari. Object-driven multi-layer scene decomposition from a single image. In *ICCV*, 2019.
- [8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014.
- [9] Roei Herzig, Moshiko Raboh, Gal Chechik, Jonathan Berant, and Amir Globerson. Mapping images to scene graphs with permutation-invariant structured prediction. In *NeurIPS*, 2018.
- [10] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, 2017.
- [11] Seunghoon Hong, Dingdong Yang, Jongwook Choi, and Honglak Lee. Inferring semantic layout for hierarchical text-to-image synthesis. In *CVPR*, 2018.
- [12] Shi-Min Hu, Fang-Lue Zhang, Miao Wang, Ralph R. Martin, and Jue Wang. Patchnet: A patch-based image representation for interactive library-driven image editing. *ACM Trans. Graph.*, 32, 2013.
- [13] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.
- [14] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, pages 1125–1134, 2017.
- [15] Seong Jae Hwang, Sathya N Ravi, Zirui Tao, Hyunwoo J Kim, Maxwell D Collins, and Vikas Singh. Tensorize, factorize and regularize: Robust visual relationship learning. In *CVPR*, pages 1014–1023, 2018.
- [16] Youngjoo Jo and Jongyoul Park. Sc-fegan: Face editing generative adversarial network with user’s sketch and color. *arXiv preprint arXiv:1902.06838*, 2019.
- [17] Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. In *CVPR*, 2018.
- [18] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*, 2017.
- [19] Justin Johnson, Andrej Karpathy, and Li Fei-Fei. Densecap: Fully convolutional localization networks for dense captioning. In *CVPR*, pages 4565–4574, 2016.
- [20] J. Johnson, R. Krishna, M. Stark, L. Li, D. A. Shamma, M. S. Bernstein, and L. Fei-Fei. Image retrieval using scene graphs. In *CVPR*, 2015.
- [21] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [22] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [23] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *IJCV*, 123(1):32–73, 2017.
- [24] Guillaume Lample, Neil Zeghidour, Nicolas Usunier, Antoine Bordes, Ludovic Denoyer, et al. Fader networks: Manipulating images by sliding attributes. In *NeurIPS*, pages 5967–5976, 2017.
- [25] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017.
- [26] Yitong Li, Zhe Gan, Yelong Shen, Jingjing Liu, Yu Cheng, Yuexin Wu, Lawrence Carin, David Carlson, and Jianfeng Gao. Storygan: A sequential conditional gan for story visualization. *arXiv preprint arXiv:1812.02784*, 2018.
- [27] Yikang Li, Wanli Ouyang, Bolei Zhou, Jianping Shi, Chao Zhang, and Xiaogang Wang. Factorizable net: an efficient subgraph-based framework for scene graph generation. In *ECCV*, pages 335–351, 2018.
- [28] Yikang Li, Wanli Ouyang, Bolei Zhou, Kun Wang, and Xiaogang Wang. Scene graph generation from objects, phrases and region captions. In *ICCV*, pages 1261–1270, 2017.
- [29] Cewu Lu, Ranjay Krishna, Michael Bernstein, and Li Fei-Fei. Visual relationship detection with language priors. In *ECCV*, 2016.
- [30] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.
- [31] Alejandro Newell and Jia Deng. Pixels to graphs by associative embedding. In *NeurIPS*, pages 2171–2180, 2017.
- [32] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *ICML*, pages 2642–2651, 2017.
- [33] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *ICML*, 2017.
- [34] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *CVPR*, 2019.
- [35] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016.
- [36] Mengshi Qi, Weijian Li, Zhengyuan Yang, Yunhong Wang, and Jiebo Luo. Attentive relational networks for mapping

- images to scene graphs. *arXiv preprint arXiv:1811.10696*, 2018.
- [37] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.
- [38] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396*, 2016.
- [39] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, pages 91–99, 2015.
- [40] Mohammad Amin Sadeghi and Ali Farhadi. Recognition using visual phrases. In *CVPR*, 2011.
- [41] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NeurIPS*, 2016.
- [42] Rakshith R Shetty, Mario Fritz, and Bernt Schiele. Adversarial scene editing: Automatic object removal from weak supervision. In *NeurIPS*, pages 7717–7727, 2018.
- [43] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [44] Wei Sun and Tianfu Wu. Image synthesis from reconfigurable layout and style. In *ICCV*, October 2019.
- [45] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *NeurIPS*, pages 4790–4798, 2016.
- [46] Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *ICML*, pages 1747–1756, 2016.
- [47] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *CVPR*, pages 8798–8807, 2018.
- [48] Danfei Xu, Yuke Zhu, Christopher Choy, and Li Fei-Fei. Scene graph generation by iterative message passing. In *CVPR*, 2017.
- [49] Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. Attribute2image: Conditional image generation from visual attributes. In *ECCV*, pages 776–791, 2016.
- [50] Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. Graph r-cnn for scene graph generation. In *ECCV*, pages 670–685, 2018.
- [51] Shunyu Yao, Tzu Ming Hsu, Jun-Yan Zhu, Jiajun Wu, Antonio Torralba, Bill Freeman, and Josh Tenenbaum. 3d-aware scene manipulation via inverse graphics. In *NeurIPS*, 2018.
- [52] Raymond A Yeh, Chen Chen, Teck Yian Lim, Alexander G Schwing, Mark Hasegawa-Johnson, and Minh N Do. Semantic image inpainting with deep generative models. In *CVPR*, pages 5485–5493, 2017.
- [53] Guojun Yin, Lu Sheng, Bin Liu, Nenghai Yu, Xiaogang Wang, Jing Shao, and Chen Change Loy. Zoom-net: Mining deep feature interactions for visual relationship recognition. In *ECCV*, pages 322–338, 2018.
- [54] Rowan Zellers, Mark Yatskar, Sam Thomson, and Yejin Choi. Neural motifs: Scene graph parsing with global context. In *CVPR*, pages 5831–5840, 2018.
- [55] Fangneng Zhan, Hongyuan Zhu, and Shijian Lu. Spatial fusion gan for image synthesis. *arXiv preprint arXiv:1812.05840*, 2018.
- [56] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiao lei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *ICCV*, pages 5907–5915, 2017.
- [57] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.
- [58] Zizhao Zhang, Yuanpu Xie, and Lin Yang. Photographic text-to-image synthesis with a hierarchically-nested adversarial network. In *CVPR*, 2018.
- [59] Bo Zhao, Bo Chang, Zequn Jie, and Leonid Sigal. Modular generative adversarial networks. In *ECCV*, 2018.
- [60] Bo Zhao, Lili Meng, Weidong Yin, and Leonid Sigal. Image generation from layout. 2019.
- [61] Yinan Zhao, Brian L. Price, Scott Cohen, and Danna Gurari. Guided image inpainting: Replacing an image region by pulling content from another image. *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1514–1523, 2018.
- [62] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *ECCV*, pages 597–613, 2016.
- [63] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017.

6. Appendix

In the following, we provide additional results, as well as full details about the implementation and training of our method. Code and data splits for future benchmarks will be released in the project web-page¹.

6.1. More Qualitative Results

Relationship changes Figure 7 illustrates in more detail our method’s behavior during relationship changes. We investigate how the bounding box placement and the image generation of an object changes when one of its relationships is altered. We compare results between auto-encoding mode and modification mode. The bounding box coordinates are masked in both cases so that the model can decide where to position the target object depending on the relationships. In auto-encoding mode, the predicted boxes (red) end up in a valid location for the original relationship, while in the altered setup, the predicted boxes respect the changed relationship, *e.g.* in auto mode, the person remains on the horse, while in modification mode the box moves beside the horse.

Spatial distribution of predicates Figure 8 visualizes the heatmaps of the ground truth and predicted bounding box distributions per predicate. For every triplet (*i.e.* subject - predicate - object) in the test set we predict the subject and object bounding box coordinates \hat{x}_i . From there, for each triplet we extract the relative distance between the object and subject centers, which are then grouped by predicate category. The plot shows the spatial distribution of each predicate. We observe similar distributions, in particular for the spatially well-constrained relationships, such as *wears*, *above*, *riding*, etc. This indicates that our model has learned to accurately localize new (predicted) objects in relation to objects already existing in the scene.

User interface video This supplement also contains a video, demonstrating a user interface for interactive image manipulation. In the video one can see that our method allows multiple changes in a given image. <https://he-dhamo.github.io/SIMSG/>

Comparison Figure 9 presents qualitative samples of our method and a comparison to [1] for the auto-encoding (a) and object removal task (b). We adapt [1] for object removal by removing a node and its connecting edges from the input graph (same as in ours), while the visual features of the remaining nodes (coming from our source image) are used to reconstruct the rest of the image. We achieve similar results for the auto-encoding, even though our method is

not specifically trained for the fully-generative task. As for object removal, our method performs generally better, since it is intended for direct manipulation on an image. For a fair comparison, in our experiments, we train [1] on Visual Genome. Since Visual Genome lacks segmentation masks, we disable the mask discriminator. For this reason, we expect lower quality results than presented in the original paper (trained on MS-COCO with mask supervision and simpler scene graphs).

6.2. Ablation study on CLEVR

Tables 3 and 4 provide additional results on CLEVR, namely for the image reconstruction and manipulation tasks. We observe that the version of our method with a SPADE decoder outperforms the other models in the reconstruction setting. As for the manipulation modes, our method clearly dominates for relationship changes, while the performance for other changes is similar with the baseline.

6.3. Datasets

CLEVR [18]. We generate 21,310 pairs of images which we split into 80% for training, 10% for validation and 10% for testing. Each data pair illustrates the same scene under a specific change, such as position swapping, addition, removal or changing the attributes of the objects. The images are of size $128 \times 128 \times 3$ and contain n random objects ($3 \leq n \leq 7$) with random shapes and colors. Since there are no graph annotations, we define predicates as the relative positions {in front of, behind, left of, right of} of different pairs of objects in the scene. The generated dataset includes annotated information of scene graphs, bounding boxes, object classes and object attributes.

Visual Genome (VG) [23]. We use the VG v1.4 dataset with the splits as proposed in [17]. The training, validation and test set contain namely 80%, 10% and 10% of the dataset. After applying the pre-processing of [17] the dataset contains 178 object categories and 45 relationship types. The final dataset after processing comprises 62,565 train, 5,506 val, and 5,088 test images with graphs annotations. We evaluate our models with GT scene graphs on all the images of the test set. For the experiments with predicted scene graphs (P), an image filtering takes place (*e.g.* no objects are detected), therefore the evaluation is performed in 3874 images from the test set. We observed relationship duplicates in the dataset and we empirically found that it does not affect the image generation task. However, it leads to ambiguity on modification time (when tested with GT graphs) once we change only one of the duplicate edges. Therefore, we remove such duplicates once one of them is edited.

¹<https://he-dhamo.github.io/SIMSG/>

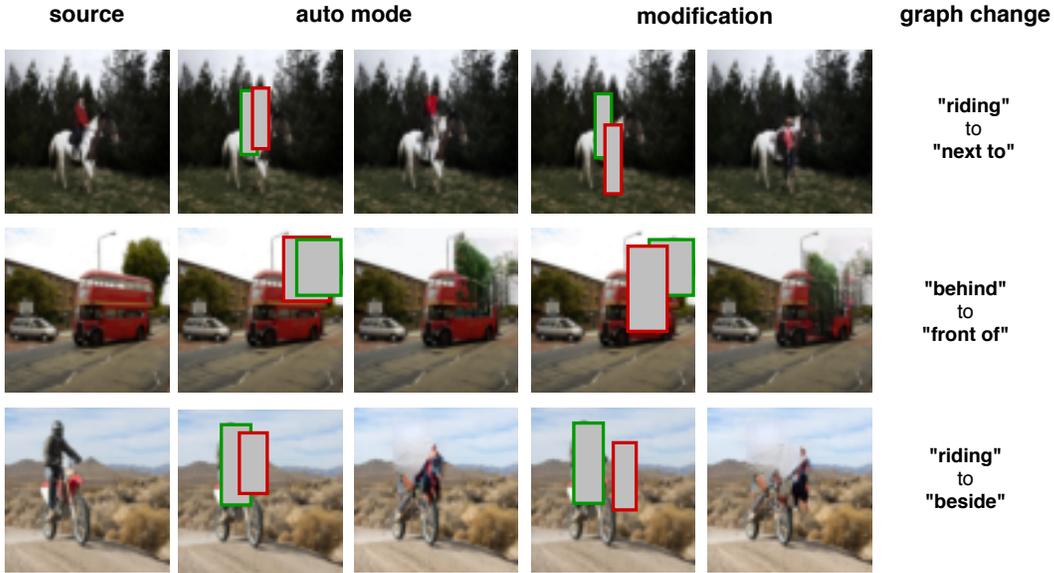


Figure 7: **Re-positioning tested in more detail.** We mask the bounding box x_i of an object and generate a target image in two modes. We choose a relationship that involves this object. In auto-mode (left) the relationship is kept unchanged. In modification mode, we change the relationship. Red: Predicted box for the auto-encoded or altered setting. Green: ground truth bounding box for the original relationship.

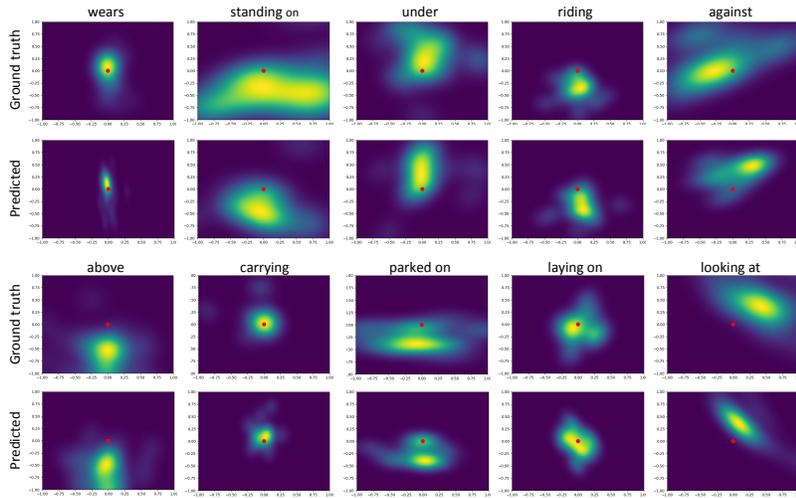


Figure 8: **Heatmaps generated from object and subject relative positions for selected predicate categories.** The object in each image is centered at point $(0, 0)$ and the relative position of the subject is calculated. The heatmaps are generated from the relative distances of centers of object and subject. Top: Ground truth boxes. Bottom: our predicted boxes (after masking the location information from the graph representation and letting it be synthesized).

6.4. Implementation details

6.4.1 Image \rightarrow scene graph

A state-of-the-art scene graph prediction network [27] is used to acquire scene graphs for the experiments on VG.

We use their publicly available implementation² to train the model. The data used to train the network is pre-processed following [6], resulting in a typically used subset of Visual Genome (sVG) that includes 399 object and 24 predicate categories. We then split the data as in [17] to avoid overlap

²<https://github.com/yikang-li/FactorizableNet>

Method	Decoder	All pixels				RoI only	
		MAE ↓	SSIM ↑	LPIPS ↓	FID ↓	MAE ↓	SSIM ↑
Image Resolution		64 × 64					
Fully-supervised	CRN	6.74	97.07	0.035	5.34	9.34	93.49
Ours (GT) w/o ϕ_i	CRN	7.96	97.92	0.016	4.52	14.36	81.75
Ours (GT) w/ ϕ_i	CRN	6.15	98.50	0.008	3.73	10.47	88.53
Ours (GT) w/o ϕ_i	SPADE	4.25	98.79	0.009	3.75	9.67	87.13
Ours (GT) w/ ϕ_i	SPADE	2.73	99.35	0.002	3.42	5.42	94.16
Image Resolution		128 × 128					
Fully-supervised	CRN	9.83	97.36	0.061	4.42	12.38	91.94
Ours (GT) w/o ϕ_i	CRN	14.82	96.85	0.041	8.09	20.59	74.71
Ours (GT) w/ ϕ_i	CRN	14.47	96.93	0.038	8.36	19.56	75.25
Ours (GT) w/o ϕ_i	SPADE	9.26	98.27	0.029	3.21	15.74	79.81
Ours (GT) w/ ϕ_i	SPADE	5.39	99.18	0.007	1.17	8.32	89.84

Table 3: **Image reconstruction on CLEVR.** We report the results using ground truth scene graphs (GT).

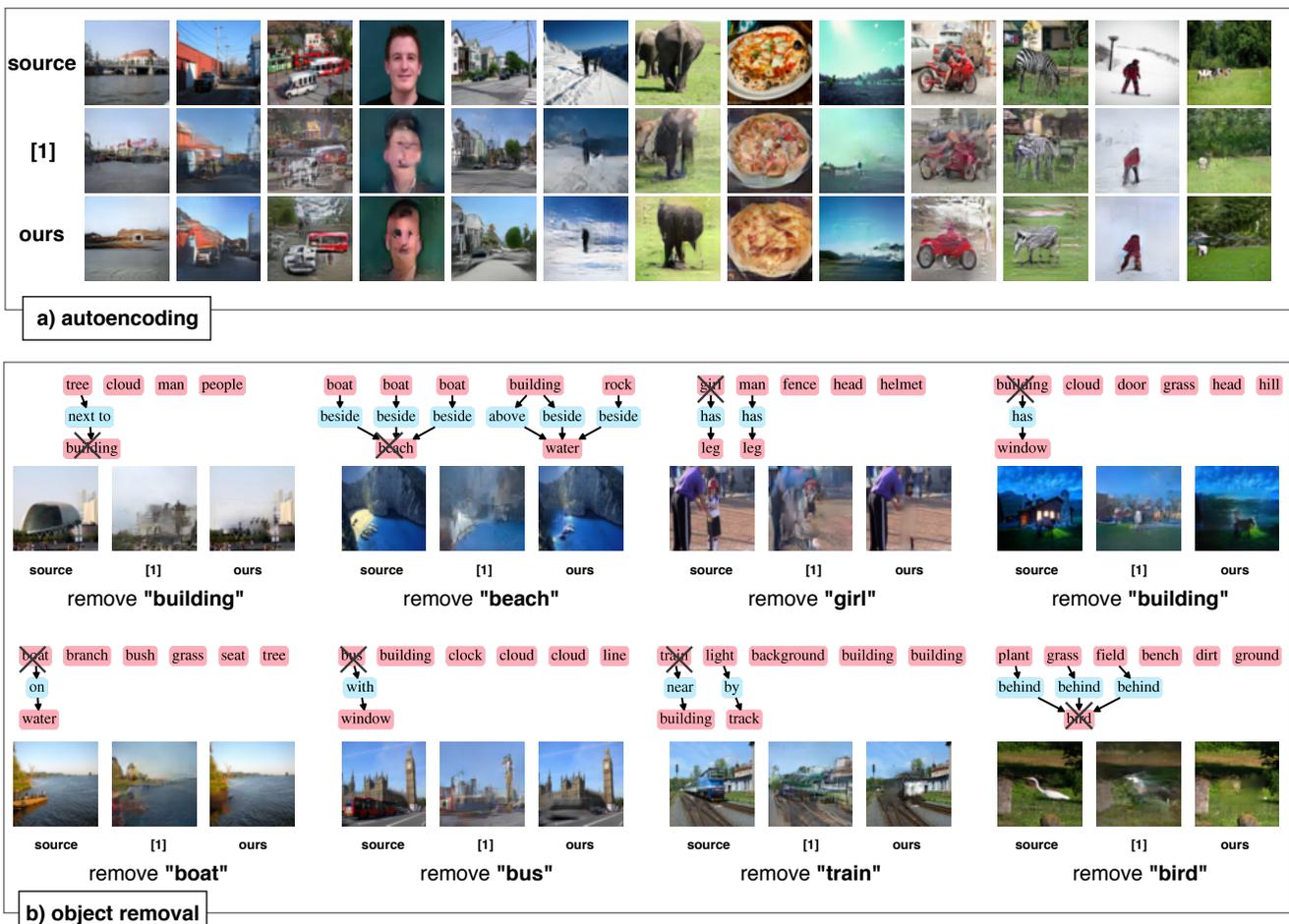


Figure 9: **Qualitative results comparing ours CRN and [1]** a) Fully-generative setting b) Object removal

in the training data for the image manipulation model. We train the model for 30 epochs with a batch size of 8 images

using the default settings from [27].

Method	Decoder	All pixels			RoI only	
		MAE ↓	SSIM ↑	LPIPS ↓	MAE ↓	SSIM ↑
Image Resolution		64 × 64				
Change Mode		Addition				
Fully-supervised	CRN	6.57	98.60	0.013	7.68	97.72
Ours (GT) w/ ϕ_i	CRN	7.88	96.93	0.027	9.79	95.10
Ours (GT) w/ ϕ_i	SPADE	4.96	97.45	0.026	6.13	96.86
Change Mode		Removal				
Fully-supervised	CRN	4.52	98.60	0.006	5.53	97.17
Ours (GT) w/ ϕ_i	CRN	5.67	97.13	0.026	7.02	96.41
Ours (GT) w/ ϕ_i	SPADE	3.45	97.32	0.022	3.88	98.09
Change Mode		Replacement				
Fully-supervised	CRN	6.64	97.76	0.015	7.33	97.11
Ours (GT) w/ ϕ_i	CRN	8.24	96.96	0.025	9.29	96.02
Ours (GT) w/ ϕ_i	SPADE	5.88	97.43	0.023	6.56	97.48
Change Mode		Relationship changing				
Fully-supervised	CRN	9.76	93.91	0.111	17.51	83.24
Ours (GT) w/ ϕ_i	CRN	10.09	93.50	0.0678	14.91	86.17
Ours (GT) w/ ϕ_i	SPADE	8.11	93.75	0.069	13.01	86.99
Image Resolution		128 × 128				
Change Mode		Addition				
Fully-supervised	CRN	9.72	97.57	0.031	10.61	94.09
Ours (GT) w/ ϕ_i	CRN	13.77	96.44	0.048	13.21	91.05
Ours (GT) w/ ϕ_i	SPADE	7.79	97.89	0.040	7.57	96.18
Change Mode		Removal				
Fully-supervised	CRN	6.15	98.72	0.014	7.27	95.58
Ours (GT) w/ ϕ_i	CRN	11.75	97.21	0.052	11.55	92.34
Ours (GT) w/ ϕ_i	SPADE	4.48	98.54	0.042	4.60	97.68
Change Mode		Replacement				
Fully-supervised	CRN	10.49	97.57	0.035	11.23	95.09
Ours (GT) w/ ϕ_i	CRN	16.38	96.14	0.052	14.74	91.98
Ours (GT) w/ ϕ_i	SPADE	10.25	97.51	0.041	9.98	96.14
Change Mode		Relationship changing				
Fully-supervised	CRN	13.91	95.26	0.169	21.49	82.46
Ours (GT) w/ ϕ_i	CRN	16.61	94.60	0.128	19.21	85.24
Ours (GT) w/ ϕ_i	SPADE	11.62	95.76	0.125	14.01	89.15

Table 4: **Image manipulation on CLEVR.** We report the results for different categories of modifications.

6.4.2 Scene graph → image

SGN architecture details. The learned embeddings of the object c_i and predicate r_i both have 128 dimensions. We create the full representation of each object o_i by concatenating c_i together with the bounding box coordinates x_i (top, left, bottom, right) and the visual features (n=128) corresponding to the cropped image region defined by the bounding box. The features are extracted by a VGG-16 architecture [43]

followed by a 128-dimensional fully connected layer.

During training, to hide information from the network, we randomly mask the visual features ϕ_i and/or object coordinates x_i with independent probabilities of $p_\phi = 0.25$ and $p_x = 0.35$.

The SGN consists of 5 layers. τ_e and τ_n are implemented as 2-layer MLPs with 512 hidden and 128 output units. The last layer of the SGN returns the outputs; the node features (s=128), binary masks (16 × 16) and bounding box coordin-

ates by 2-layer MLP with a hidden size of 128 (which is needed to add or re-position objects).

CRN architecture details. The CRN architecture consists of 5 cascaded refinement modules, with the output number of channels being 1024, 512, 256, 128 and 64 respectively. Each module consists of two convolutions (3×3), each followed by batch normalization [13] and leaky Relu. The output of each module is concatenated with a down-sampled version of the initial input to the CRN. The initial input is the concatenation of the predicted layout and the masked image features. The generated images have a resolution of 64×64 .

SPADE architecture details. The SPADE architecture used in this work contains 5 residual blocks. The output number of channels is namely 1024, 512, 256, 128 and 64. In each block, the layout is fed in the SPADE normalization layer, to modulate the layer activations, while the image counterpart is concatenated with the result. The global discriminator D_{global} contains two scales.

The object discriminator in both cases is only applied on the image areas that have changed, *i.e.* have been in-painted.

Full-image branch details. The image regions that we randomly mask during training are replaced by Gaussian noise. Image features are extracted using 32 convolutional filters (1×1), followed by batch normalization and Relu activation. Additionally, a mask is concatenated with the image features that is 1 in the regions of interest (noise) and 0 otherwise, so that the areas to be modified are easier for the network to identify.

Training settings. In all experiments presented in this paper, the models were trained with Adam optimization [21] with a base learning rate of 10^{-4} . The weighting values for different loss terms in our method are shown in Table 5. The batch size for the images in 64×64 resolution is 32, while for 128×128 is 8. All objects in an image batch are fed at the same time in the object-level units, *i.e.* SGN, visual feature extractor and discriminator.

All models on VG were trained for 300k iterations and on CLEVR for 40k iterations. Training on an Nvidia RTX GPU, for images of size 64×64 takes about 3 days for Visual Genome and 4 hours for CLEVR.

6.5. Failure cases

In the proposed image manipulation task we have to restrict the feature encoding to prevent the encoder from “copying” the whole RoI, which is not desired if, for instance, we want to re-position non-rigid objects, e.g. from sitting to standing. While the model is able to retain general

Loss factor	Weight CRN	Weight SPADE
λ_g	0.01	1
λ_o	0.01	0.1
λ_a	0.1	0.1
λ_b	10	50
λ_f	-	10
λ_p	-	10

Table 5: Loss weighting values

appearance information such as colors and textures, it is true that, as a side effect some visual properties of modified objects are not recovered. For instance, the color of the green object in Figure 10 a) is preserved but not the material.

The model does not adapt unchanged areas of the image as a consequence of a change in the modified parts. For example, shadows or reflections do not follow the re-positioned objects, if those are not nodes of the graph and explicitly marked as changing subject by the user, Figure 10 b).

In addition, similarly to other methods evaluated on Visual Genome, the quality of some close objects remains limited, *e.g.* close-up of people eating, Figure 10 c). Also, having a node `face` on animals, typically gives them a human face.

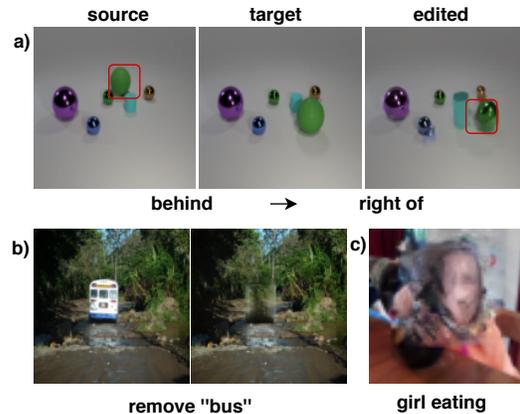


Figure 10: Illustration of failure cases.