Appendix - Semantic Image Manipulation Using Scene Graphs

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-page[1].

1. More Qualitative Results

Relationship changes Figure 1 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 2 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 3 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).

[1] https://he-dhamo.github.io/SIMSG/

2. Ablation study on CLEVR

Tables 1 and 2 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.

3. Datasets

CLEVR [5]. 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) [7]. We use the VG v1.4 dataset with the splits as proposed in [4]. The training, validation and test set contain namely 80%, 10% and 10% of the dataset. After applying the pre-processing of [4] 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

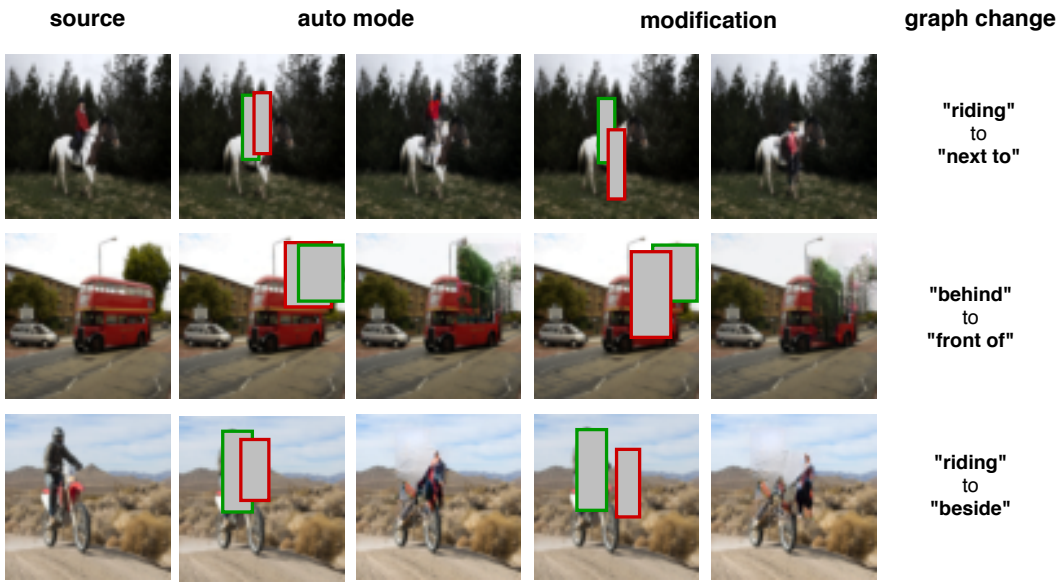

Figure 1. **Re-positioning tested in more detail.** We mask the bounding box x_i of an object and generate a target image in two modes. We choose a relationship that involves this object. In auto-mode (left) the relationship is kept unchanged. In modification mode, we change the relationship. Red: Predicted box for the auto-encoded or altered setting. Green: ground truth bounding box for the original relationship.

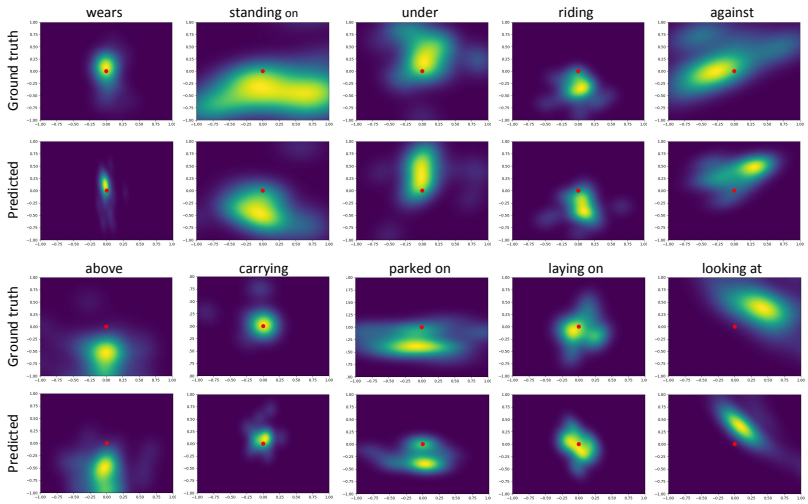

Figure 2. **Heatmaps generated from object and subject relative positions for selected predicate categories.** The object in each image is centered at point $(0, 0)$ and the relative position of the subject is calculated. The heatmaps are generated from the relative distances of centers of object and subject. Top: Ground truth boxes. Bottom: our predicted boxes (after masking the location information from the graph representation and letting it be synthesized.

our models on the images of 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.

4. Implementation details

4.1. Image → scene graph

A state-of-the-art scene graph prediction network [8] is used to acquire scene graphs for the experiments on VG. We use their publicly available implementation[2] to train the model. The data used to train the network is pre-processed following [2], resulting in a typically used subset of Visual

[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 1. **Image reconstruction on CLEVR.** We report the results using ground truth scene graphs (GT).

Figure 3. **Qualitative results comparing ours CRN and [1] a)** Fully-generative setting **b)** Object removal

Genome (sVG) that includes 399 object and 24 predicate categories. We then split the data as in [4] to avoid overlap 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 [8].

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 2. **Image manipulation on CLEVR.** We report the results for different categories of modifications.

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 corresponding to the cropped image region defined by the bounding box. The features are extracted by a VGG-16 architecture [9] followed by a 128-dimensional fully connected layer. A linear projection layer then projects the total object representation down to 128-d.

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 (128-d), binary masks (16×16) and bounding box coordi-

nates 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 [3] 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, while the image counterpart is concatenated with the result. The global discriminator D_{global} contains two scales.

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 [6] with a base learning rate of 10^{-4}. The weighting values for different loss terms in our method are shown in Table 3. The batch size for the images is 32. 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 takes about 3 days for VG and 4 hours for CLEVR.

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 3. Loss weighting values

4.3. 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 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 4 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 4 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 4 c). Also, having a node "face" on animals, typically gives them a human face.

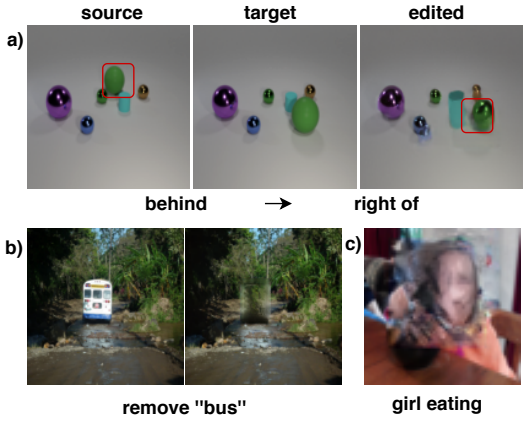

Figure 4. Illustration of failure cases.

References

[1] Oron Ashual and Lior Wolf. Specifying object attributes and relations in interactive scene generation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4561–4569, 2019. 1, 3

[2] Bo Dai, Yuqi Zhang, and Dahua Lin. Detecting visual relationships with deep relational networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3076–3086, 2017. 2

[3] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 5

[4] Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. In *CVPR*, 2018. 1, 3

[5] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 1

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[7] 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. *International Journal of Computer Vision*, 123(1):32–73, 2017. 1

[8] 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 *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 335–351, 2018. 2, 3

[9] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 4