Supplementary

Specifying Object Attributes and Relations in Interactive Scene Generation

Oron Ashual
Tel Aviv University
oron.ashual@gmail.com

Lior Wolf
Tel Aviv University and Facebook AI Research
wolf@cs.tau.ac.il, wolf@fb.com

A. A full version of the paper images

Sample results of our 256x256 model are shown in Fig. 1, using test images from the COCO-stuff datasets. Each row presents the scene layout, the ground truth image from which the layout was extracted, our method's results, where the object attributes present a random archetype and the location attributes are zeroed ($l_i = 0$), our results when using the ground truth layout of the image (including masks and bounding boxes), our results where the appearance attributes of each object are copied from the ground truth image and the location vectors are zero, and our results where the location attributes coarsly desribe the objects' locations and the appearance attributes are randomly selected from the archetypes. In addition, we present the result of the baseline method of [1] at the 64x64 resolution for which a model was published.

As can be seen, our model produces realistic results across all settings, which are visually more pleasing than the baseline method. Using ground truth location and appearance attributes, the resulting image better matches the test image.

Fig. 2 presents samples obtained when sampling the appearance attributes. In each case, for all i, $l_i = 0$ and the objects appearance embedding a_i is sampled uniformly between the archetypes. This results in a considerable visual diversity.

Fig. 3 presents results in which the appearance is fixed to the mean appearance vector for all objects of that class and the location attribute vectors l_i are sampled from the Gaussian distributions mentioned above. In almost all cases, the generated images are visually pleasing. In some cases, the location attributes sampled are not compatible with generating a photo realistic image. Note, however, that in our method, the default value for l_i is zero and not a random vector.

The ability of our method to copy the appearance of an existing image object is demonstrated in Fig. 4. In each example, we generate the same test scene graph, while varying a single object with accordance to four different options extracted from images unseen during training. Despite the variability of the appearance that is presented in the four sources, the generated images mostly maintain their visual quality. These results are presented at a resolution of 256x256, which is the default resolution for our GUI. At this resolution, the system processes a graph in 16.3ms.

B. Network architecture

The graph convolutional network used is the same as the one used in [1], which was modified in order to support the added node information. We concatenate the embedding of the objects to the location attributes creating vectors in R^{128+35}. These vectors, together with the relation embedding is feed-forward into a fully-connected layer which results in a vector embedding in $\mathbb{R}^{128}$ for each of the objects and each of the relations. The network then follows the architecture of [1], using a shared symmetric function to calculate the object vector from all the relations it participate in. Our graph convolution has overall five layers.

To describe the rest of the networks, we follow a semi-conventional shorthand notation for defining architectures. Let C_k denote a Convolution layer of k filters with a kernel size of $7x7$ and a stride of 1, followed by instance normalization [2] and a ReLU activation function. Similarly, we use D_k to a layer which uses a stride of 2, reflection padding, and k filters. In addition, we use B_k to denote a $3x3$ upsample-convolution-BatchNormalization-ReLU layer with k filters and a stride and padding of 1. We use V_k to define Residual blocks with two $3x3$ convolutional layers, both with k filters. Moreover, U_k denotes a layer with k filters of size $3x3$ and a fractional stride of 0.5, followed by instance normalization. CB_k denotes a stride-2 k-filter, $4x4$ convolution followed by batch normalization. GA denotes a global average pooling layer. Fully connected layers with k hidden units followed by a ReLU activation are denoted by L_k. The ReLU is not applied to the L_k layer, if it is the top layer.

The discriminators call for an even more elaborate terminology. Let C_{i-k-o} denote a 3x3 Convolution layer with i input channels and k output filters, a stride of o and a padding of 1. In addition, LR denotes Leaky-ReLU with negative slope of 0.2, IN denotes Instance Normalization, and AP_k denotes a $3x3$ Average Pool stride 2 and padding of 1. Also, let C_{k-s} denote a Convolution layer with k filters, stride of s, kernel size of $4x4$, and a padding of size 2. D_{k-s} denotes a $4x4$ Convolution-InstanceNorm-LeakyReLU layer with k filters, a stride of s padding of 2 and a LeakyReLU with a negative slope of 0.2.

The different components of the networks can be described as:

M B_{192}, B_{192}, B_{192}, B_{192}, B_{192}, B_{192}, C_1, Sigmoid activation

B L_{128}, L_{512}, L_4

A CB_{64}, CB_{128}, CB_{256}, GA, L_{192}, L_{64}, L_{32}

R C_{c+32}, D_{128}, D_{256}, D_{512}, D_{1024}, V_{1024}, V_{1024}, V_{1024}, V_{1024}, V_{1024}, V_{1024}, V_{1024}, V_{1024}, U_{512}, U_{256}, U_{128}, U_{64}, C_3

D_{mask} C_{1-64-2}, LR, $C^*_{(c+64)-128-1}$, IN, LR, $C_{128-1-1}$, AP

D_{image} $C_{(c+32)-2}$, LR, D_{64-2}, D_{128-2}, D_{256-1}, C_{512-1}

D_{object} CB_{64},CB_{128},$C4_{256}$, GA, L_{1024}, L_1

(*Concatenating the c_i data in D_{image} is done at the third layer)

Acknowledgement

This project has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (grant ERC CoG 725974).

References

[1] Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 3

[2] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. 1

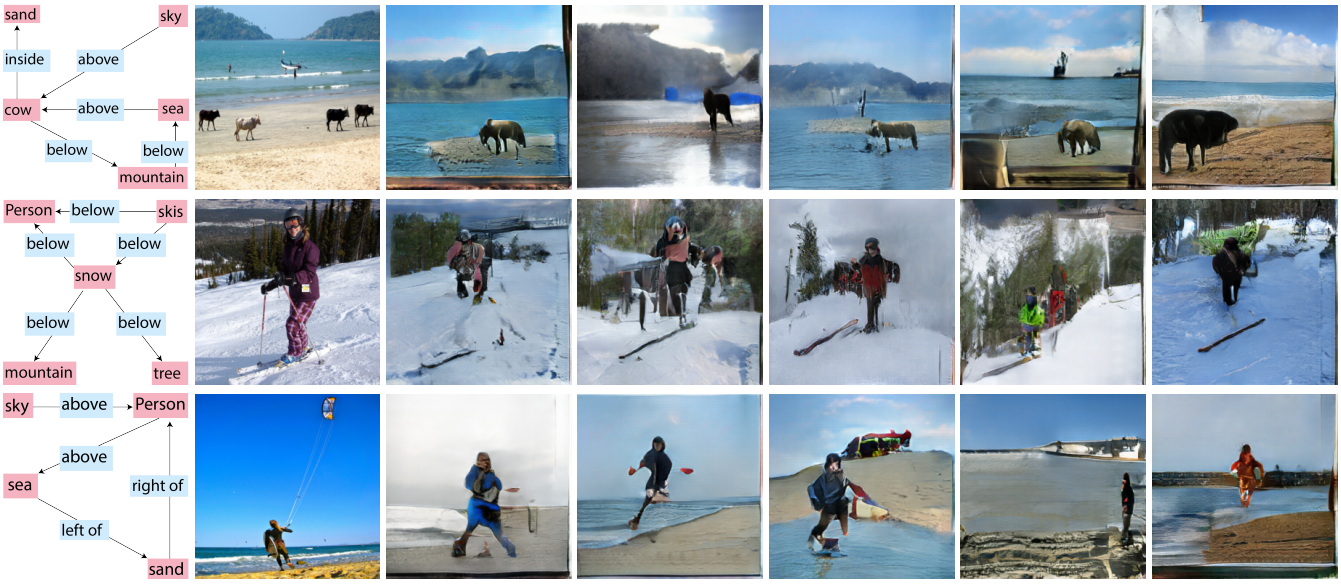
[3] Bo Zhao, Lili Meng, Weidong Yin, and Leonid Sigal. Image generation from layout. *CoRR*, abs/1811.11389, 2018. 3

(a) (b) (c) (d) (e) (f) (g)

Figure 1. Image generation based on a given scene graph. Each row is a different example. (a) the scene graph, (b) the ground truth image from which the layout was extracted, (c) our results when we used the ground truth layout of the image, similarly to [3], (d) our method's results, where the appearance attributes present a random archetype and the location attributes coarsely describe the ground truth bounding box,(e) our results when we use the ground truth image to generate the appearance attributes and $l_i = 0$, (f) our results where the location attributes are zeroed $l_i = 0$ and the appearance attributes are sampled from the archetypes and (g) the result of [1].

(a) (b) (c) (d) (e) (f) (g)

Figure 2. The diversity obtained when keeping the location attributes l_i fixed at zero and sampling different appearance archetypes. (a) the scene graph, (b) the ground truth image from which the layout was extracted, (c–g) generated images.



(a) (b) (c) (d) (e) (f) (g)

Figure 3. The diversity obtained when keeping the appearance vectors fixed and sampling from the location distribution. (a) the scene graph, (b) the ground truth image from which the layout was extracted, (c–g) generated images.

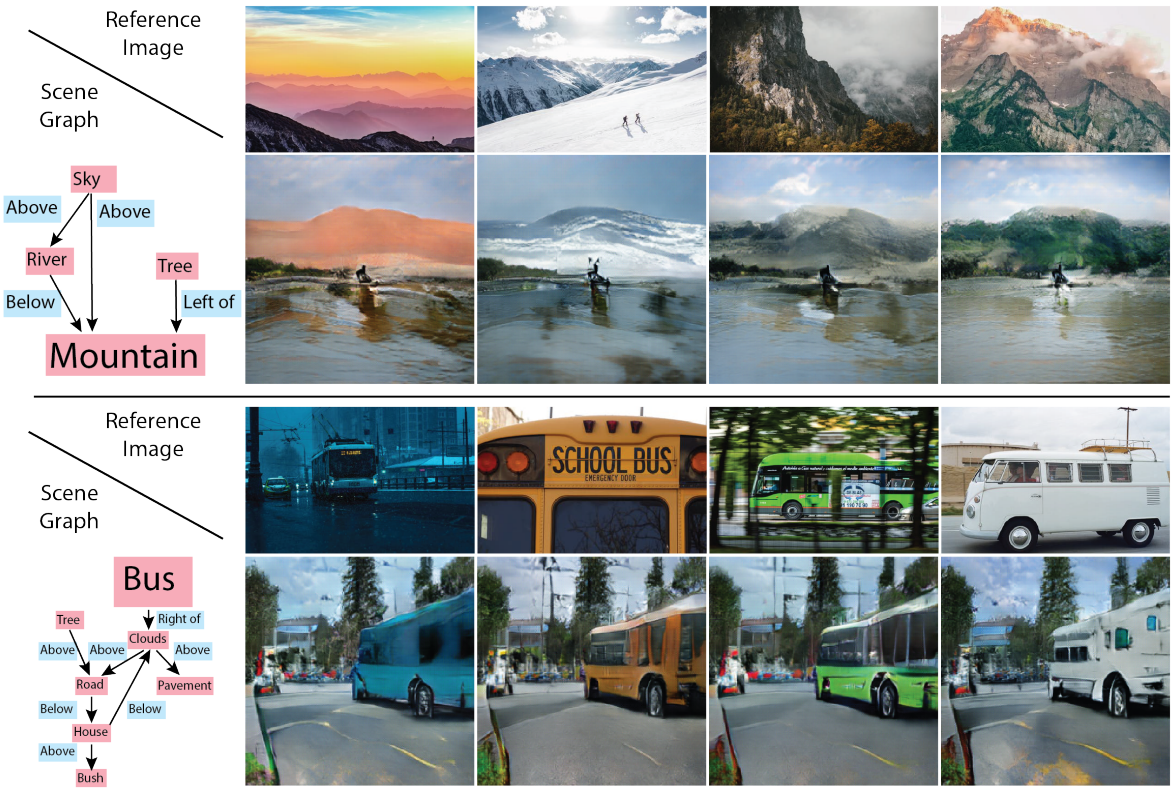


Figure 4. Duplicating an object's appearance in the generated image. In each of the three samples, images are created based on the scene graph, such that the appearance is taken from one of four unrelated images. In the first sample, the sky's appearance is copied. In the second, the mountain changes. In the third sample, the appearance of the bus object is copied from one of four images.