Scene Designer: a Unified Model for Scene Search and Synthesis from Sketch

Leo Sampaio Ferraz Ribeiro[1], Tu Bui[2], John Collomosse[2, 3], and Moacir Ponti[1]

[1]ICMC, Universidade de São Paulo – São Carlos/SP, Brazil
[2]CVSSP, University of Surrey – Guildford, Surrey, UK
[3]Adobe Research, Creative Intelligence Lab – San Jose, CA, USA

Supplementary Material

1. Designing QuickDrawCOCO

For each object in a COCO-stuff [1] scene, we randomly select a QuickDraw[1] sketch from the same class and replace the object crop. To do so, a map from QuickDraw classes to COCO classes was made; Out of 345 sketch classes in QuickDraw, 119 have correspondents in the COCO set, while from the 172 classes in COCO-stuff, 92 have sketch correspondents. The mapping is not 1:1 because Quick-Draw is "more specific" than COCO-stuff, e.g. 'school bus' and 'bus' are both classes in QuickDraw, but both are mapped to 'bus' under COCO-stuff's set. The full class map is shown in Table 3.

We call the objects with sketch correspondents *sketchable* objects; any non-*sketchable* object is excluded from the list of objects at the preprocessing stage. We keep however "objects" from categories that we called *materials*; while those do not have sketched correspondents, they remain necessary for background generation when synthesizing the final image; QuickDrawCOCO has 56 *materials* that can compose the background. The list of *materials* is in Table 2.

Finally, the dataset is also restricted by number of *sketchable* objects, from 1 to 8, and those objects need to occupy at least 5% of the scene area. Given this setup, QuickDraw-COCO is composed of 111,112 training scenes (images, plus the full set of possible synthesized sketch compositions), 2,788 test scenes and 1,907 validation scenes. The test and validation splits where split from COCO's initial test set (2,048 for valiation and 2,952 for testing) and then filtered per our requirements.

2. Architecture and Preprocessing Details

The crops are scaled to 96x96 in preprocessing, as well as their values normalized to the [-1, 1] floating point range. The QuickDraw sketches are from the set used by SketchRNN and are rasterized to 96x96 images from the stroke3 format. The output is the 256D SR for search and the 256x256 semantic layout.

[1]https://github.com/googlecreativelab/quickdraw-dataset

To create our **Object-Level Representation (OLR)**, the sketches and images crops go through two separate MobileNet, the specific model used was trained on ImageNet with 96x96 image size and is available through Tensorflow/Keras[2]. The output of the network goes is average pooled to decrease the dimentionality and then the model splits into two steps: one fc layer with softmax activation for each domain classifies the inputs while a set of 2 shared fc layers computes the OLR used in the triplet loss. The specifics of those layers are in Table 1.

Our Graph Neural Network (GNN) encodes into a **Constrained-Correlated Representation (CCR)** the Scene Graph (SG) where each node is represented by the object's corresponding OLR. Our GNN has 6 layers, each built with hidden dimensionality of 512 (the output of the first internal fc layer), output of 256 and all activations done via LeakyRelu with $\alpha = 0.3$. Our code was based on Johnson *et al.*'s [2] implementation[3] of a GNN for SGs.

Finally, our Transformer-like architecture takes the CCR and computes both the single **Scene Representation (SR)** and the **Freely-Correlated Representations (FCR)** for each object. Our grid-based positional encoding follows traditional encoders and is defined as:

$$p_{i,j} = \begin{cases} sin(\frac{i}{10000^{j/256}}) & \text{if } j \text{ is even} \\ cos(\frac{i}{10000^{(j-1)/256}}) & \text{if } j \text{ is odd} \end{cases} \quad (1)$$

where $i \in 0, 1, ..., 24$ is the position in the grid while $j \in 0, 1, ..., 255$ is the position in the vector (which is 256D long). Our Transformer has 3 attention layers that use multi-head attention (16 heads), a hidden dimension of 512D, dropout with rate 0.1 and Layer Normalisation with $\epsilon = 1e - 6$. Our implementation was based on the official TensorFlow code[4].

To generate scenes using the *materials* listed in Table 2, a semantic layout using them can be selected from images ranked high in the SBIR process or a user could make their own background semantic layouts in a similar fashion to the SPADE demonstration [3].

[2]https://www.tensorflow.org/api_docs/python/tf/keras/applications/MobileNet
[3]https://github.com/google/sg2im
[4]https://www.tensorflow.org/tutorials/text/transformer

3. Training Details

As mentioned before, training is done in three stages. In the **first stage** we train only the OLR representation and the total loss function is:

$$\mathcal{L}_{OLR} = \mathcal{L}_{tri} + \mathcal{L}_{CCE} \tag{2}$$

where both losses were defined in Sec. 3.1 in the paper. The negative crops used for $\mathcal{L}_{tri}$ are random crops from other classes. This stage is trained with a batch size of 128 samples from QuickDraw (sketches) and COCO-stuff (crops from the images) for 100k iterations.

At the **second stage** all parts of Scene Designer are trained and losses used:

$$\mathcal{L}_{SD} = \mathcal{L}_{OLR} + \mathcal{L}_{G_b} + \mathcal{L}_{G_m} + \mathcal{L}_{cont} + \mathcal{L}_{CCE_f} \tag{3}$$

with each of those loss functions defined in Sec. 3.4 and 3.5 in the paper. The mask discriminator D is also trained with the LSGAN loss, opposing the one in $\mathcal{L}_{G_b}$:

$$\mathcal{L}_D = \frac{1}{2}|D(m_g, y), 1|_2 + \frac{1}{4}|D(G_m(\hat{x}_i, y))|_2 + \frac{1}{4}|D(G_m(\hat{x}_s, y))|_2 \tag{4}$$

where $\hat{x}_i$ as the FCR based on image input and $\hat{x}_s$ as the FCR based on sketch input and m_g the ground truth masks. This stage is trained with a batch size of 16 sketch+image scene pairs from QuickDrawCOCO-92c for 120k iterations.

Finally, the **third stage** uses the same losses as the second, but with a batch size of 8 samples from QuickDrawCOCO-92c and 8 samples from SketchyCOCO. We've used mixed batches as using only SketchyCOCO samples leads Scene Designer to overfit.

4. Implementing Our Baselines

We used the official code and models provided for Ashual *et al.* [4][5] and SketchyCOCO's EdgeGAN [5] model[6]. We have to note however that the EdgeGAN is only the single-object generator of SketchyCOCO's model and we trained our own Pix2Pix (also using official code[7]) to do full scene generation per Gao *et al.*'s instructions.

To run experiments on Sketchy, we used an unnoficial pytorch reimplementation that yields comparable or better results than the official paper[8] and for the CAG [6] model we did our own reimplementation. Numbers for SceneSketcher [7] and Sketch-me-that-shoe [8] were taken from Liu *et al.*'s report [7].

[5] https://www.github.com/ashual/scene_generation
[6] https://github.com/sysu-imsl/EdgeGAN
[7] https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix
[8] https://github.com/CDOTAD/SketchyDatabase

Layer	In	Out	Act.
MobileNet	(96, 96, 1/3)	(7, 7, 1024)	-
AveragePool	(7, 7, 1024)	(1024,)	-
FC^1	(1024,)	(256,)	-
FC_{class}	(256,)	(172,)	Softmax
FC^1_{olr}	(256,)	(128,)	ReLU
FC^2_{olr}	(128,)	(128,)	ReLU

Table 1. Layers that compute the Object-level Representation (OLR).

Material	Mat.	Mat.	Mat.	Mat.
ceiling-other	floor-tile	curtain	stone	sea
window-other	cardboard	counter	rock	hill
wall-concrete	wall-tile	carpet	cloth	snow
floor-marble	floor-wood	pavement	river	tree
floor-marble	wall-stone	railing	gravel	fog
ceiling-tile	floor-stone	railroad	straw	metal
structural-other	solid-other	plastic	road	net
ground-other	floor-other	banner	roof	mud
window-blind	wall-other	wall-wood	wood	mat
textile-other	solid-other	sky-other	sand	dirt
playingfield	wall-brick	clothes	other	moss
water-other	plant-other	leaves	grass	

Table 2. List of *materials* in QuickDrawCOCO. These object classes are used for background layouts.

5. More Examples

Together with this supplementary material we are including new examples of generation, search and iterative composition on both SketchyCOCO and QuickDrawCOCO-92c. They are presented in Fig. 1, 2, 3, 4, 5 and 6.

References

[1] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. Coco-stuff: Thing and stuff classes in context. In *Computer vision and pattern recognition (CVPR), 2018 IEEE conference on*. IEEE, 2018. 1

[2] Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image Generation from Scene Graphs. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1219–1228. IEEE, jun 2018. 1

[3] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 1

[4] Oron Ashual and Lior Wolf. Specifying object attributes and relations in interactive scene generation. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2019-Octob, pages 4560–4568, sep 2019. 2

[5] Chengying Gao, Qi Liu, Qi Xu, Limin Wang, Jianzhuang Liu, and Changqing Zou. Sketchycoco: Image generation from freehand scene sketches. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

COCO-stuff class	QuickDraw classes	COCO-stuff class	QuickDraw classes
chair	chair	boat	canoe; cruise ship; sailboat; speed boat
cow	cow	cake	birthday cake; cake
bench	bench	train	train
pillow	pillow	toothbrush	toothbrush
clouds	cloud	pizza	pizza
light	floor lamp; lantern; light bulb	broccoli	broccoli
table	table	motorcycle	motorbike
bear	bear	person	yoga
building-other	church; hospital;	couch	couch
airplane	airplane	refrigerator	cooler
toaster	toaster	spoon	spoon
knife	knife; sword	oven	oven; stove
fence	fence	fire hydrant	fire hydrant
cabinet	dresser	furniture-other	hot tub
baseball bat	baseball bat	sink	sink
vegetable	onion; peas; potato; string bean; aspargus	cup	coffee cup; cup; mug
		food-other	peanut; steak; bread;
horse	horse	microwave	microwave
traffic light	traffic light	waterdrops	rain
umbrella	umbrella	hot dog	hot dog
banana	banana	tv	television
wine glass	wine glass	bus	bus; school bus
fork	fork	fruit	grapes; pear; pineapple; strawberry; watermelon; blackberry; blueberry
branch	tree		
bush	bush	sports ball	soccer ball; baseball; basketball
toilet	toilet	handbag	suitcase; purse
donut	donut	cell phone	cell phone
tennis racket	tennis racquet	mountain	mountain
elephant	elephant	vase	vase
backpack	backpack	sandwich	hamburger; sandwich
bottle	wine bottle	cage	jail
car	car; police car	skyscraper	skyscraper
remote	remote control	carrot	carrot
bicycle	bicycle	wall-panel	picture frame
bed	bed	laptop	computer; laptop
bridge	bridge	bird	bird
stairs	stairs	tie	bowtie
teddy bear	teddy-bear	skateboard	skateboard
apple	apple	paper	map
book	book	sheep	sheep
house	house; barn	zebra	zebra
door-stuff	door	cat	cat
mouse	mouse	clock	wristwatch; alarm clock; clock
flower	flower	truck	ambulance; firetruck; pickup truck; van; truck
tent	tent	dog	dog
keyboard	keyboard	potted plant	house plant
stop sign	stop sign	giraffe	giraffe

Table 3. Mapping between COCO and QuickDraw *sketchable* classes. With COCO as our base, more than one QuickDraw class may be mapped to a COCO class.

[6] Tu Bui, Leo Ribeiro, Moacir Ponti, and John Collomosse. Sketching out the details: Sketch-based image retrieval using convolutional neural networks with multi-stage regression. *Computers & Graphics*, 71:77–87, 2018. 2

[7] Fang Liu, Changqing Zou, Xiaoming Deng, Ran Zuo, Yu-Kun Lai, Cuixia Ma, Yong-Jin Liu, and Hongan Wang. Scenesketcher: Fine-grained image retrieval with scene sketches. 2020. 2

[8] Qian Yu, Feng Liu, Yi-Zhe Song, Tao Xiang, Timothy M Hospedales, and Chen-Change Loy. Sketch me that shoe. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 799–807, 2016. 2

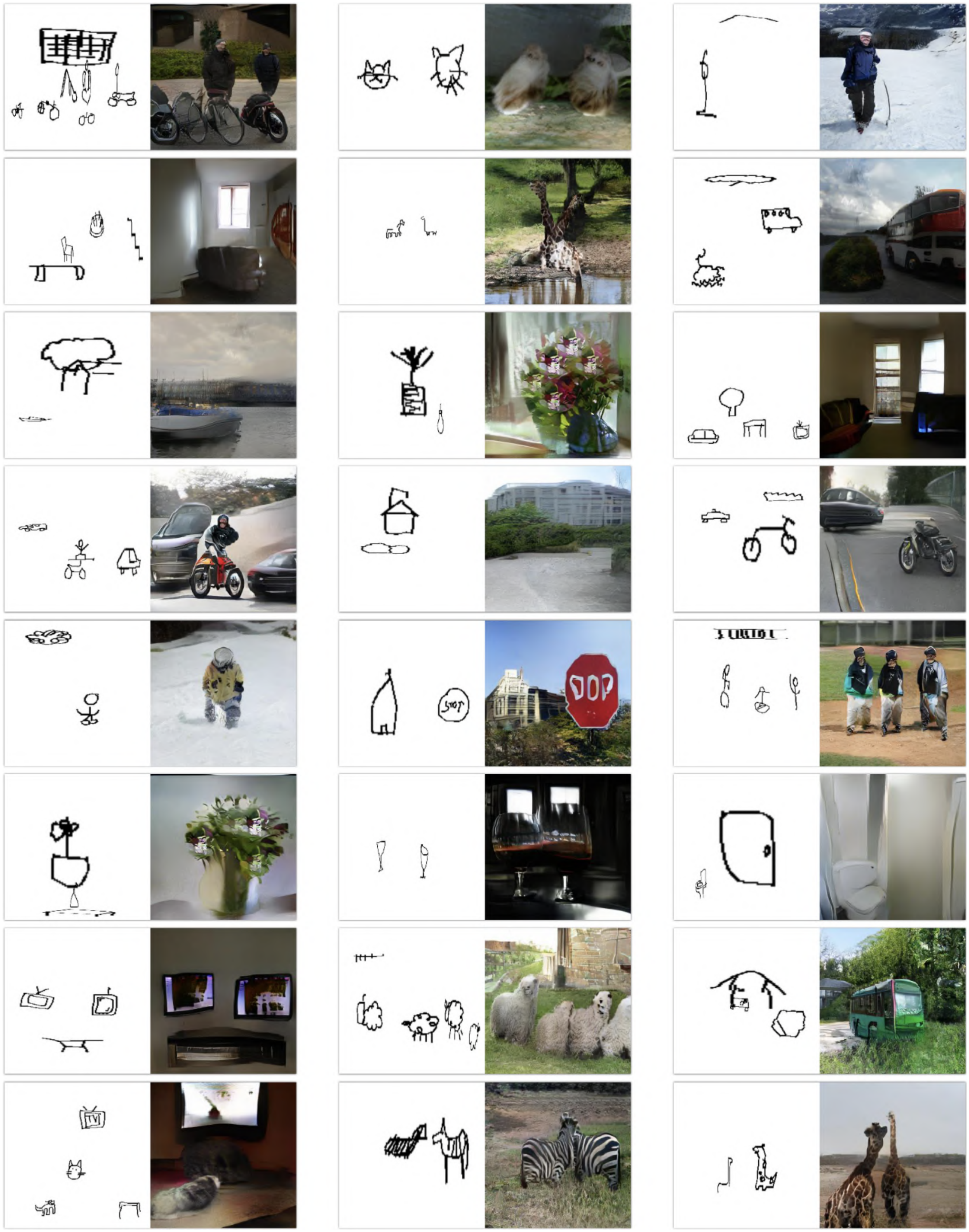


Figure 1. Generating images with Scene Designer with more QuickDrawCOCO-92c scenes

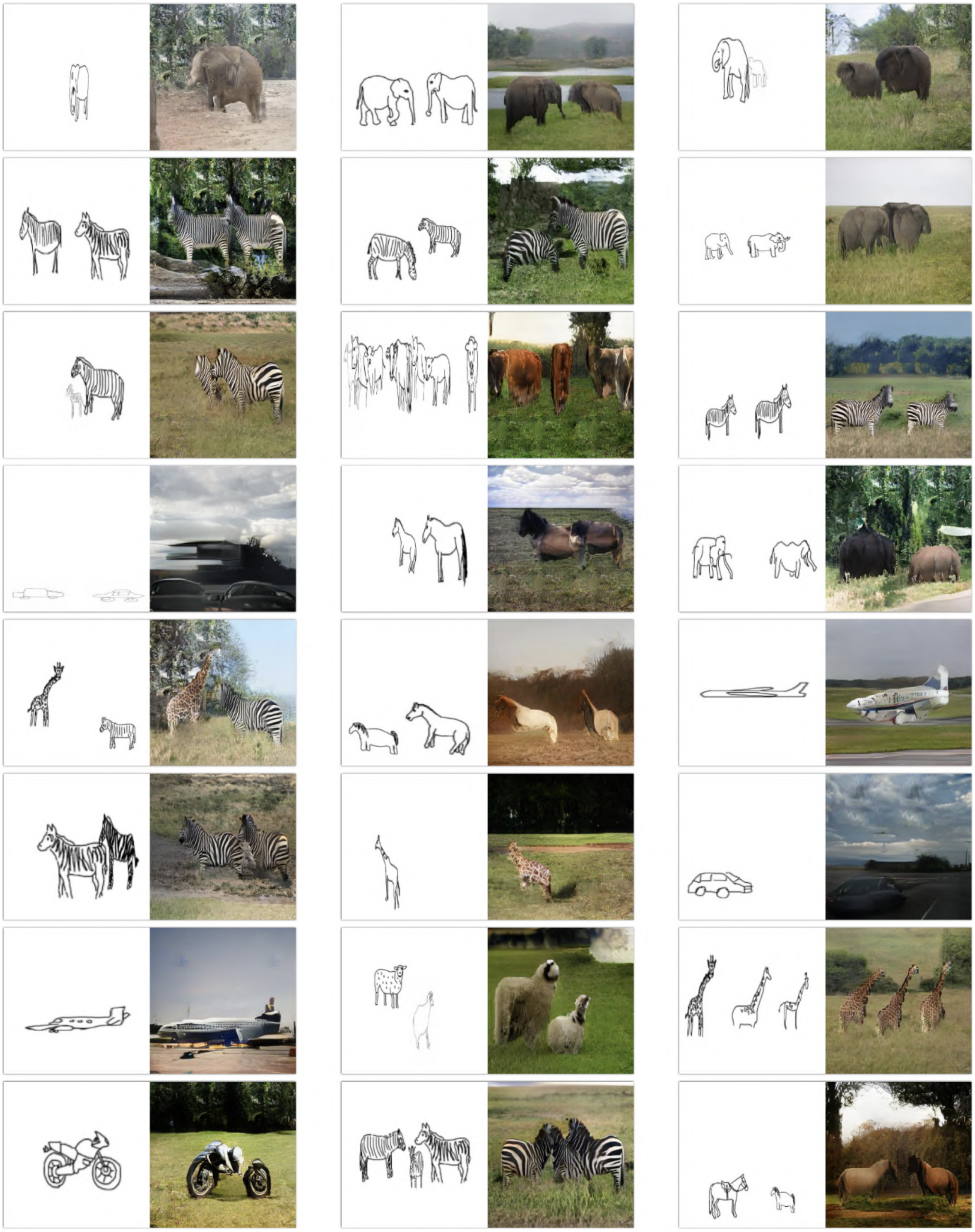


Figure 2. Generating images with Scene Designer with more SketchyCOCO scenes

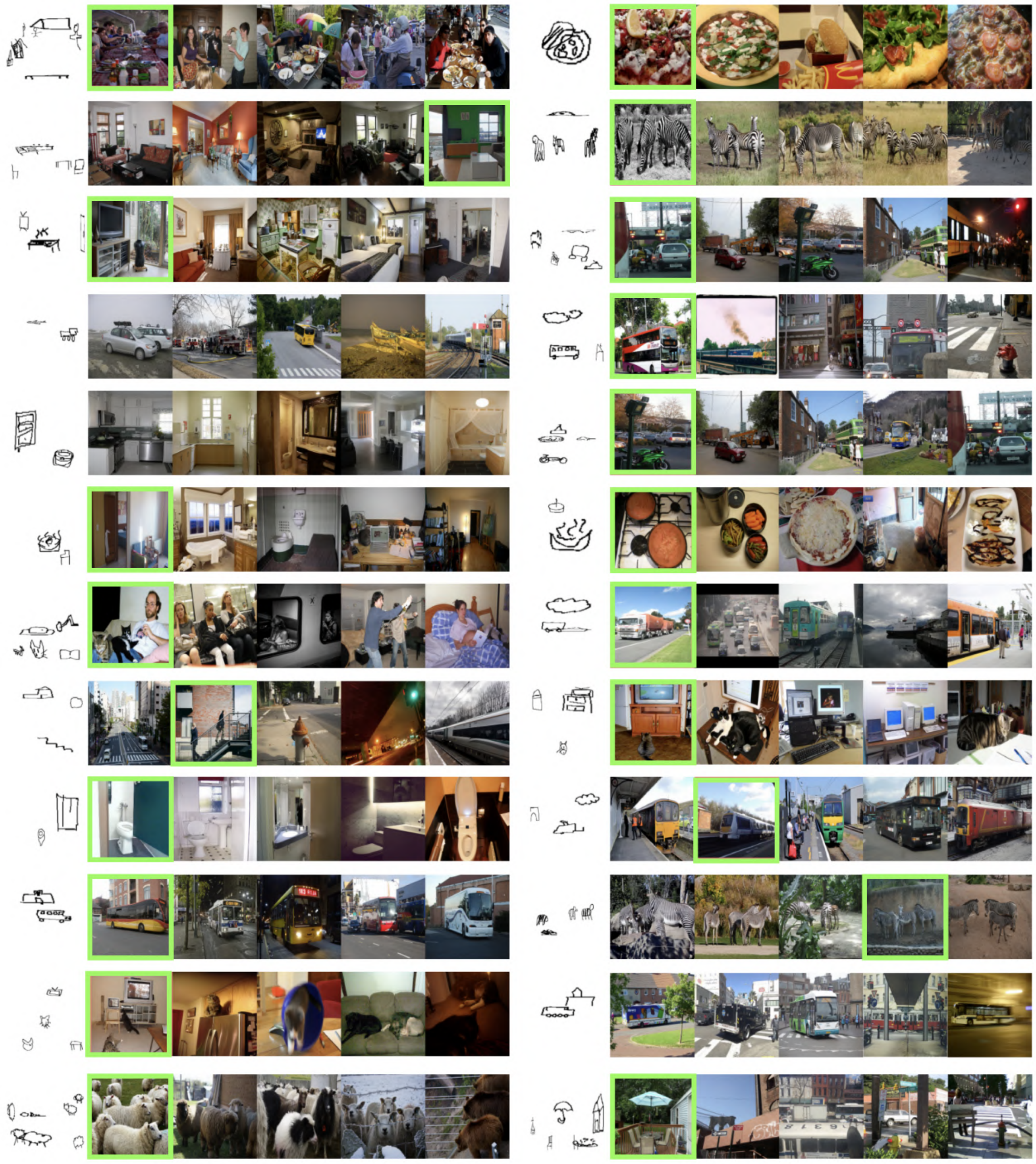


Figure 3. Compositional Sketch-based Image Retrieval results with Scene Designer on more QuickDrawCOCO-92c scenes; showing top-5 results with the correct match highlighted in green.

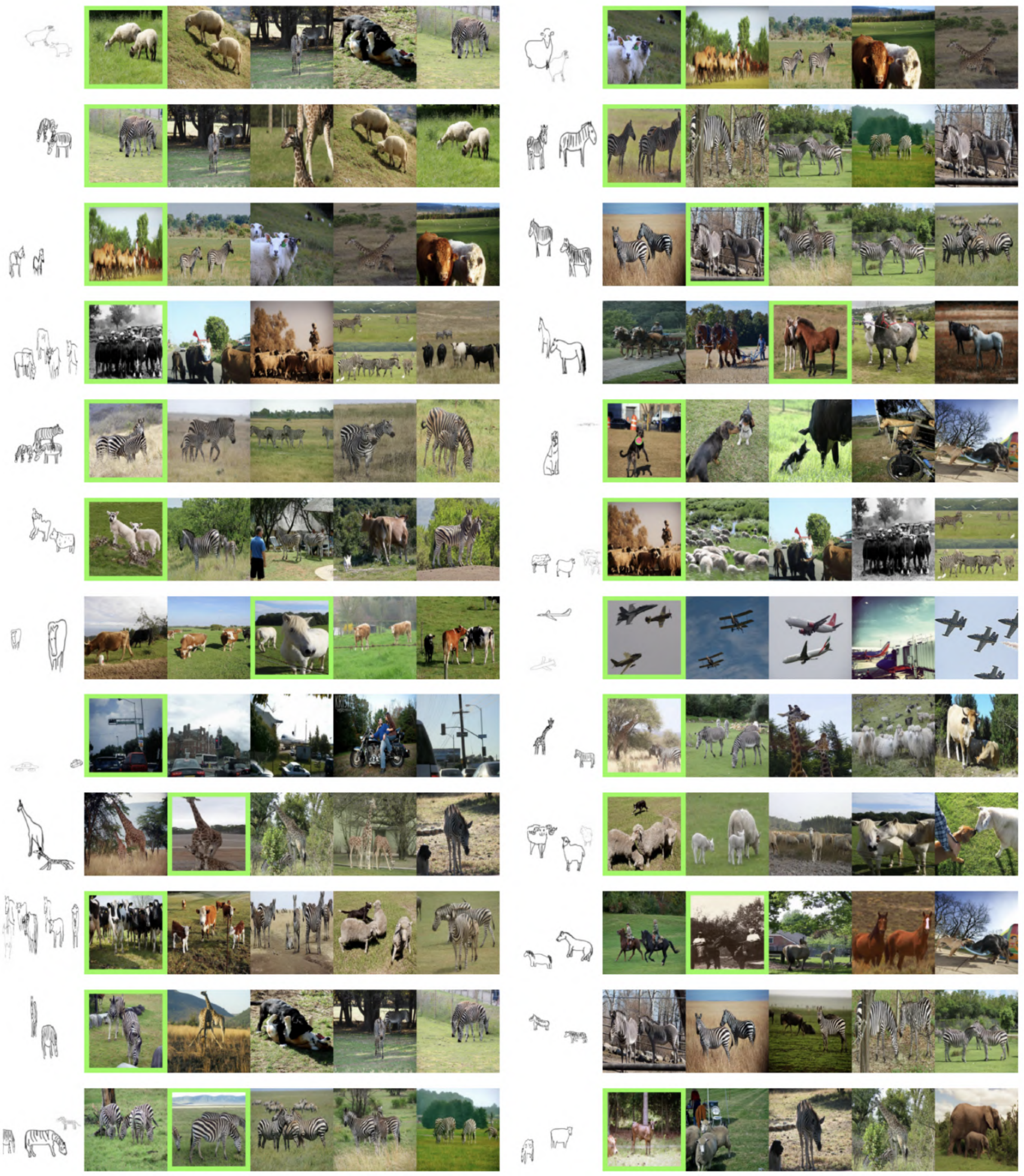
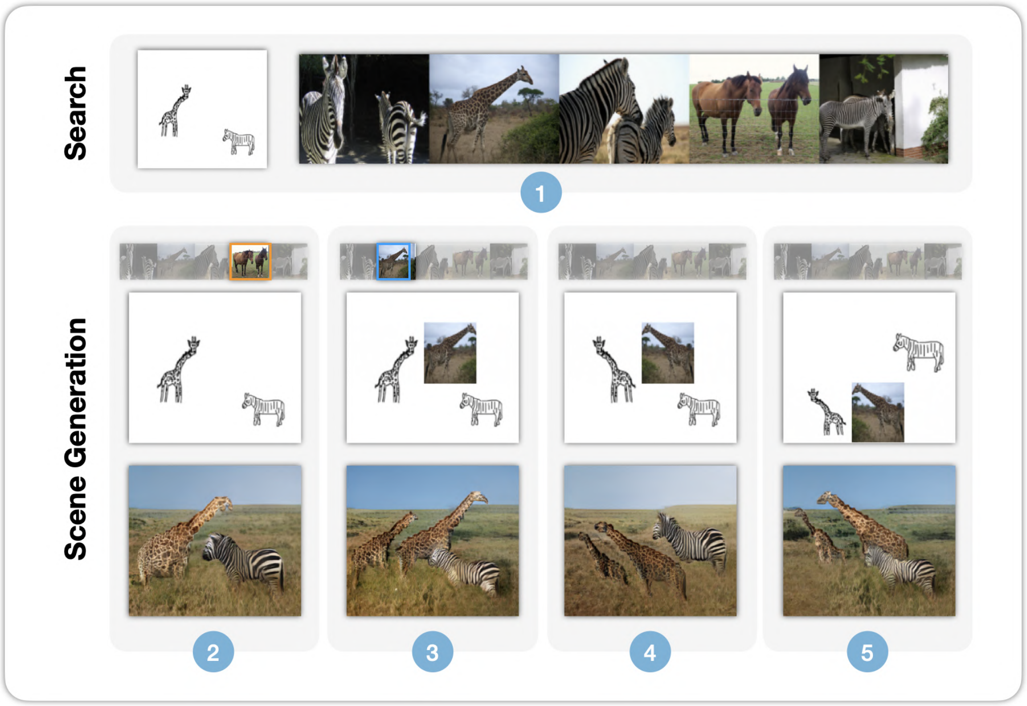


Figure 4. Compositional Sketch-based Image Retrieval results with Scene Designer on more SketchyCOCO scenes; showing top-5 results with the correct match highlighted in green.



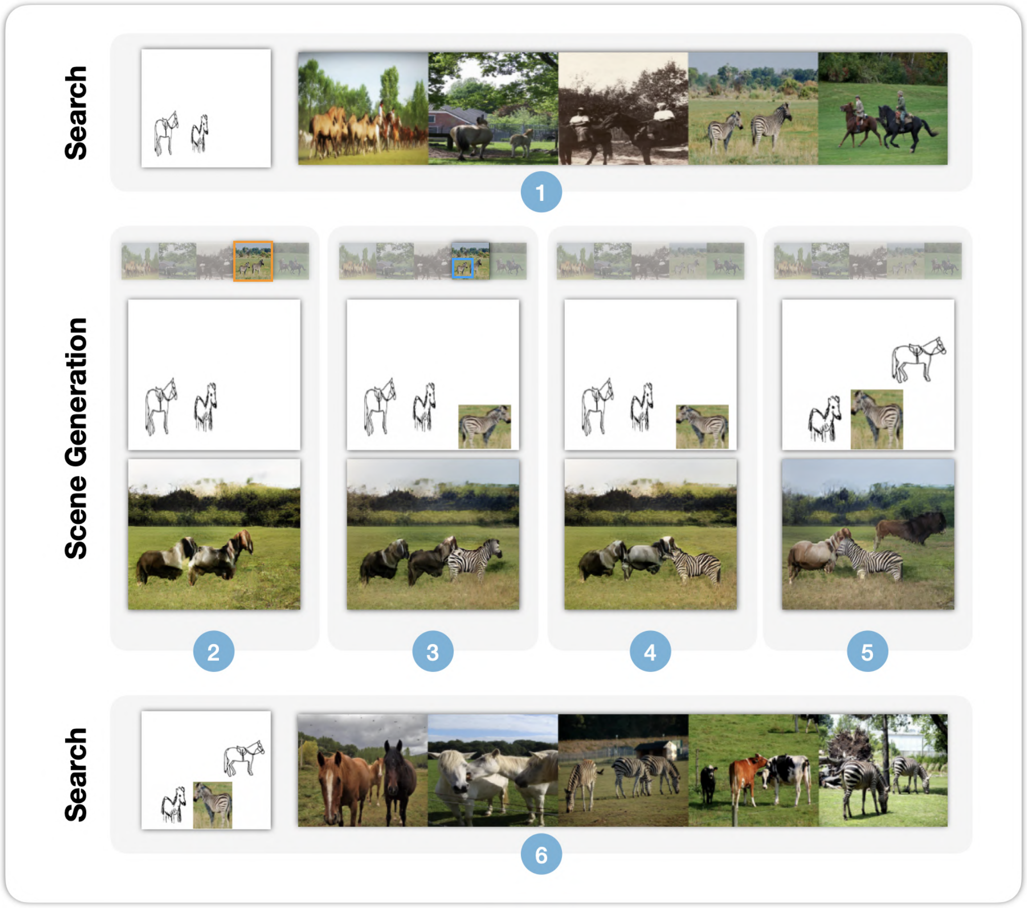


Figure 5. More examples of using Scene Designer for iterative design of image compositions using Sketchy sketches and SketchyCOCO scenes. Orange squares indicate the image's background layout was selected; Blue squares show an object crops was chosen to be added to the composition.

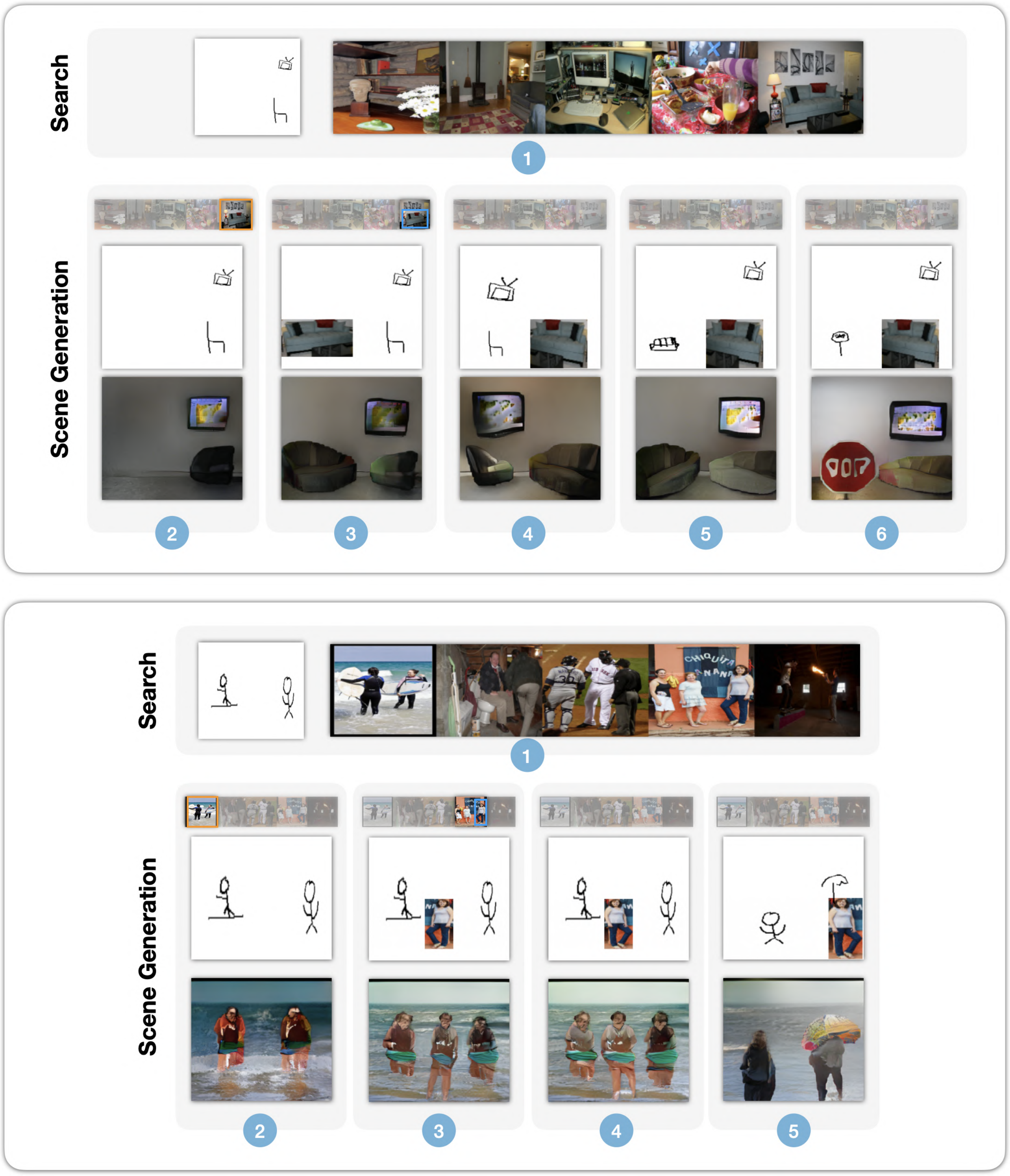


Figure 6. More examples of using Scene Designer for iterative design of image compositions using QuickDraw sketches and QuickDrawCOCO-92c scenes. Orange squares indicate the image‘s background layout was selected; Blue squares show an object crops was chosen to be added to the composition.