Supplementary materials for: Diagnostic Benchmark and Iterative Inpainting for Layout-Guided Image Generation

Appendix

In this Appendix, we provide additional ITERINPAINT experiments, including image generation with user-defined layouts, interactive image manipulation, ablation studies, and training on COCO images (Sec. A), LAYOUTBENCH-COCO details (Sec. B), examination of CLEVR GT layout accuracy (Sec. C), additional GAN baseline (Sec. D), and additional LAYOUTBENCH image samples (Sec. E).

A. Additional ITERINPAINT Experiments

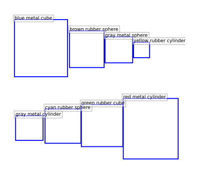
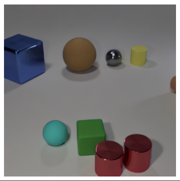
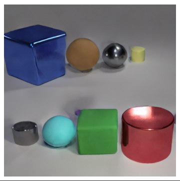
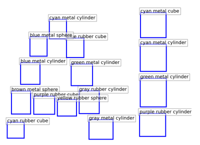
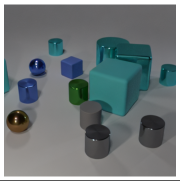
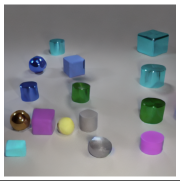
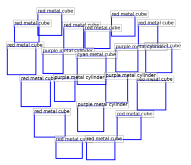
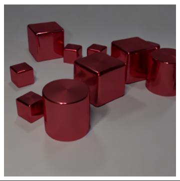
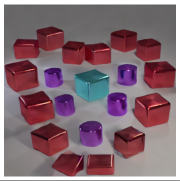
Layout	ReCo	ITERINPAINT
		
		
		

Table 7. Image generation from some user-defined layouts with ITERINPAINT and ReCo trained on CLEVR. On the leftmost column, we show three input layouts: (1) two rows of objects with different sizes, (2) 'AI' written in the text, and (3) a heart shape.

A.1. Image Generation with User-defined Layouts

Table 7 shows image generation results from user-defined layouts with CLEVR objects: (1) two rows of objects with different sizes, (2) 'AI' written in the text, and (3) a heart shape. While ReCo often fails to ignore or misplace some objects, ITERINPAINT places objects significantly more accurately. This shows the high robustness of ITERINPAINT, which can follow even more abstract and complex than the automatically generated layouts of LAYOUTBENCH.

A.2. Interactive Image Manipulation

Figure 6 shows interactive image manipulation examples. With ITERINPAINT, users can interactively remove or add objects from a given image at arbitrary locations. When removing objects, we use a prompt 'Add gray background'.

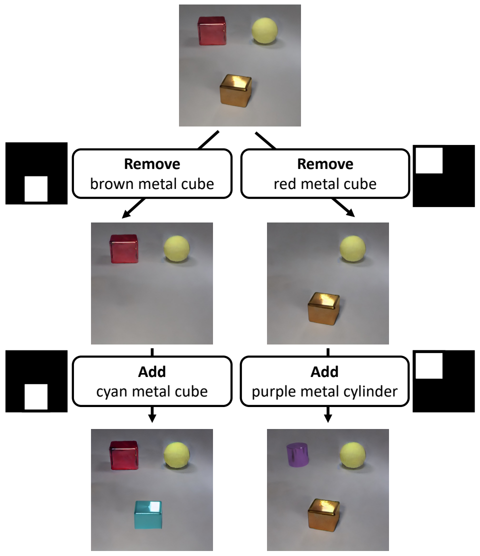


Figure 6. Interactive image manipulation with ITERINPAINT. Users can create a binary mask (white: region to update / black: region to preserve) to add or remove objects at custom locations.

A.3. ITERINPAINT Ablation Study

Pasting *vs.* Repaint. Instead of crop&paste, we experiment the repainting the entire image during the inference. As shown in Tab. 9, repainting results in -12.1% AP. Tab. 8 shows that the repaint-based update encodes/decodes the

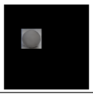
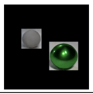
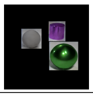
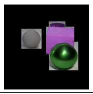
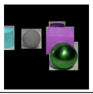
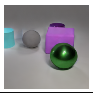
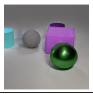
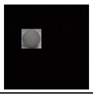
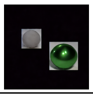
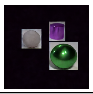
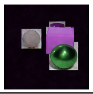
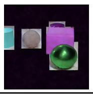
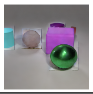
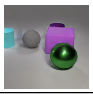
Update	Intermediate Generation						GT image
Prompts	Step 1 Add gray rubber sphere	Step 2 Add green metal sphere	Step 3 Add purple metal cylinder	Step 4 Add purple rubber cube	Step 5 Add cyan rubber cylinder	Step 6 Add gray background	
Crop&Paste (default)							
Repaint							

Table 8. Intermediate generation samples of ITERINPAINT with crop&paste (top) and repaint (bottom) based image updates.

Image Update	FG/BG Training Ratio	Generation Order	Checkpoint	LAYOUTBENCH AP
Crop & Paste	3:7	Random	SD Inpaint	36.5
Repaint				24.4 (-12.1)
	1:9			36.0 (-0.5)
	2:8			35.2 (-1.3)
	4:6			35.8 (-0.7)
	5:5			35.0 (-1.5)
	6:4			34.4 (-2.1)
	7:3			34.2 (-2.3)
	8:2			34.2 (-2.3)
	9:1			32.9 (-3.6)
	10:0 (No BG)			27.5 (-9.0)
		Top $\rightarrow$ Bottom		37.2 (+0.7)
		Bottom $\rightarrow$ Top		36.1 (-0.4)
			SD v1.4	31.4 (-5.1)

Table 9. Ablation studies on ITERINPAINT

whole image at each step and suffers from error propagation (*i.e.*, early objects get distorted with step progress).

Training task ratio for foreground & background inpainting. Sec. 4.3 in the main paper shows the LAYOUTBENCH layout accuracy with different ratios for foreground/background inpainting tasks. We found that the 3:7 ratio performs best, but other task ratios perform similarly, except for the case of 10:0, where not using background inpainting tasks results in a significant -9.0% drop in AP.

Object generation order. Because of the camera angle of CLEVR simulator, if there is an occlusion between objects, it always takes the form of an object in the bottom (front), occluding the object above it in 2D coordinates. We compare 3 generation orders for rendering objects given the spatial coordinates: top$\rightarrow$bottom, bottom$\rightarrow$top, and random. Unlike other iterative generation models that are sensitive in generation orders [1, 6], Tab. 9 shows that ITERINPAINT achieves similar AP with all 3 generation orders. The robustness in arbitrary generation order would be useful when users want to manipulate object layouts without re-rendering images from scratch.

SD Checkpoint. In our experiments, the SD 'inpainting' checkpoint [1] shows slightly better layout accuracy than v1.4 checkpoint[2], so we use the inpainting checkpoint by default for ITERINPAINT.

[1]https://huggingface.co/runwayml/stable-diffusion-inpainting

[2]https://huggingface.co/CompVis/stable-diffusion-v1-4

A.4. Training on COCO images

Although our main focus is constructing a diagnostic LAYOUTBENCH benchmark with full scene control and evaluating the layout-guided image generation models, we also test whether our proposed ITERINPAINT baseline model could also perform well on real images. We train ITERINPAINT on MS COCO [5] train2014 split, for around 100 epochs with batch size 512 (after applying gradient accumulations). In Table 10, we show some image generation samples from ReCo (COCO checkpoint) and ITERINPAINT from COCO layouts, where both models could locate objects in the correct positions. In Table 11, we show some arbitrary custom layouts with COCO objects. While both models are correct in object locations, ReCo sometimes fails to place wrong objects that are frequent in a given layout, while ITERINPAINT shows a more precise object recognition performance. Note that the original ReCo model was trained with much bigger expensive training resources (e.g., ReCo uses batch size 2048 vs. our ITERINPAINT uses batch size 512), and we leave bigger scale training and hyperparameter tuning to future works.

B. LAYOUTBENCH-COCO Details

B.1. Dataset

The LAYOUTBENCH-COCO (Sec. 5.5 in the main paper) benchmarks layout-guided image generation models in zero-shot fashion. Following the design of LAYOUTBENCH, LAYOUTBENCH-COCO measures four skills: number, position, size, and combination. The idea of number/position/size skills is essentially the same as those splits of LAYOUTBENCH, while we replace the shape skill of LAYOUTBENCH with a new 'combination' skill. The combination skill measures whether a model can generate uncommon combinations of two objects, which is a specifically interesting evaluation scenario for zero-shot models; for LAYOUTBENCH evaluation, the models finetuned on CLEVR have seen many combinations of different objects, so the combination of different objects is a less interesting problem.

For each skill, we first define 2D bounding box layouts without specifying objects. Then, we compose COCO [5]

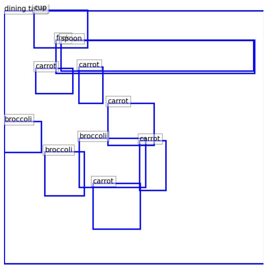
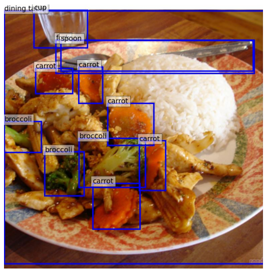
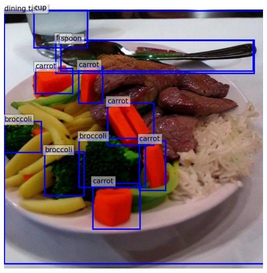
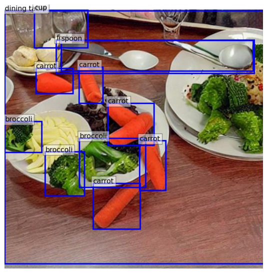
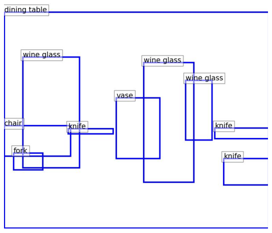
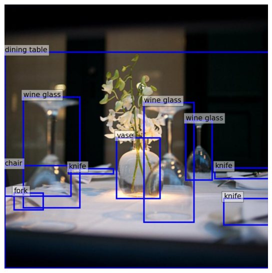
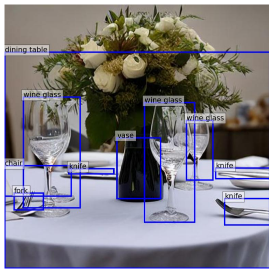
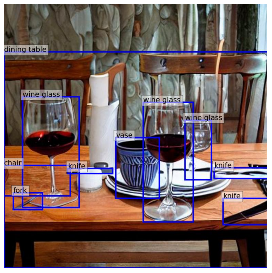
Layout	GT image	ReCo	ITERINPAINT
			
			

Table 10. Image generation samples with COCO layouts (in-distribution) from ITERINPAINT and ReCo.

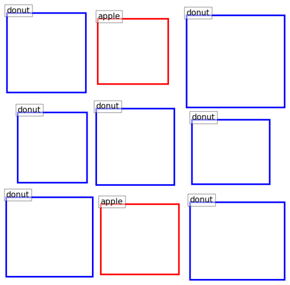
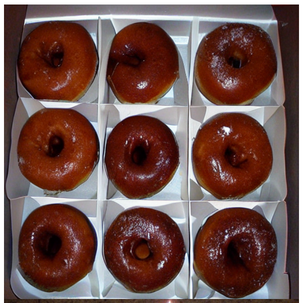
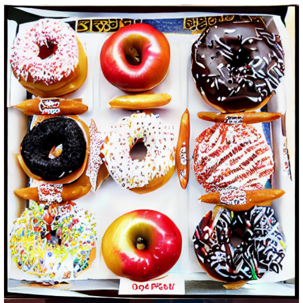
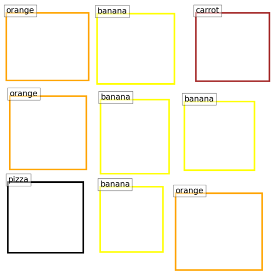
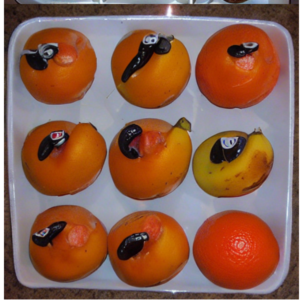
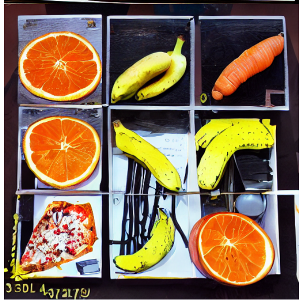
Layout	ReCo	ITERINPAINT
		
		

Table 11. Image generation samples with custom layouts with COCO objects (out-of-distribution) from ITERINPAINT and ReCo.

objects for each layout. This process ensures that the layouts have balanced distributions among objects. In total, LAYOUTBENCH-COCO provides 2,120 layouts. In Table 12, we show example layouts for each skill. In the following, we describe how we create the layouts for each skill in detail.

Skill 1: Number. We define two layouts for 2$\sim$10 objects and use 40 COCO objects, resulting in 720 total layouts (= $2 \times 9 \times 40$). We name the layouts with 2$\sim$4 and 8$\sim$10 objects as *few* and *many* splits. The layouts are paired with captions with a template "`a photo of [N] [objects]`".

Skill 2: Position. For each of *boundary* and *center* splits, we define four layouts with 40 COCO objects, resulting in 320 total layouts (= $2 \times 4 \times 40$). The layouts are

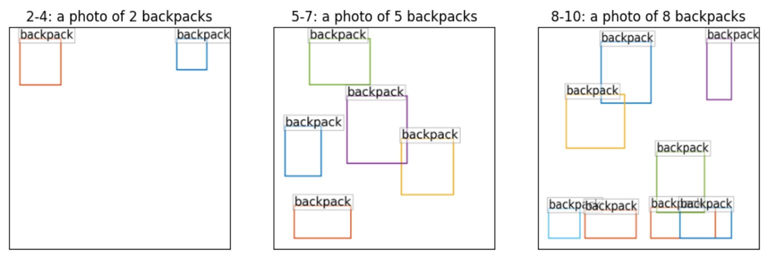
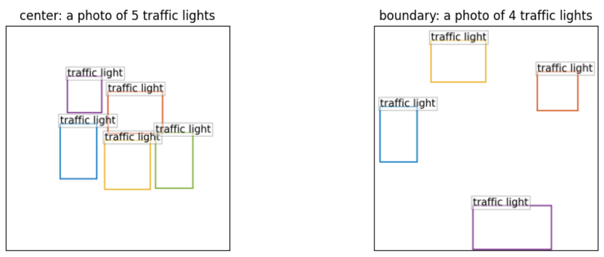
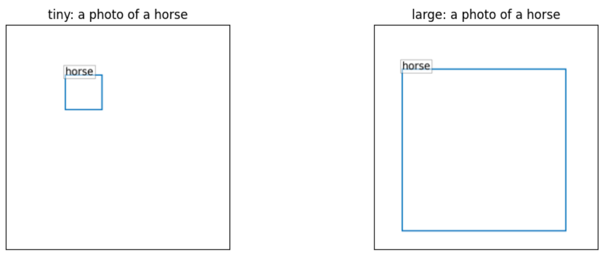
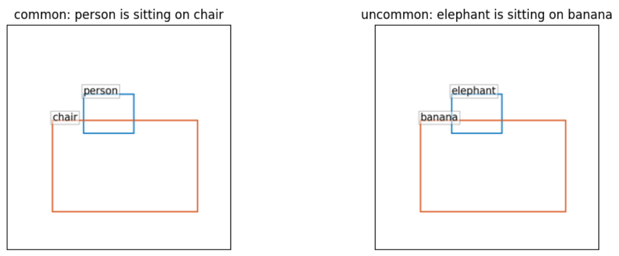
Skill 1: Number	 2-4: a photo of 2 backpacks 5-7: a photo of 5 backpacks 8-10: a photo of 8 backpacks backpack
Skill 2: Position	 center: a photo of 5 traffic lights boundary: a photo of 4 traffic lights traffic light
Skill 3: Size	
Skill 4: Combination	 common: person is sitting on chair uncommon: elephant is sitting on banana person chair elephant banana

Table 12. Example layouts and captions of the four skills of LAYOUTBENCH-COCO.

paired with captions with a template "`a photo of [N] [objects]`".

Skill 3: Size. For each of *tiny* and *large* splits, we define nine layouts with 40 COCO objects, resulting in 720 total layouts ($= 9 \times 2 \times 40$). The layouts are paired with captions with a template "`a photo of [N] [objects]`".

Skill 4: Combination. This skill measures whether a model can generate two objects that commonly or uncommonly appear in the real world. For each of the three spatial relations (holding, next to, sitting on), we define three layouts without specifying objects. For each of the three relations, we manually define 20 object pairs of COCO objects for *common* and *uncommon* splits. For example, (1) 'person sitting on chair' is more common than (2) 'elephant sitting on banana' in real life. This results in 360 total layouts ($= 2 \times 3 \times 3 \times 20$). The layouts are paired with captions with a template "`[objA] [relation] [objB]`".

B.2. Qualitative Examples

In Table 13 and Table 14, we show sample images generated by different layout-guided image generation models from LAYOUTBENCH-COCO layouts.

C. CLEVR GT Layout Accuracy

In Table 2, the AP on CLEVR GT images (60.7) is not as high as that of LAYOUTBENCH GT images (90.7). This is because of the noise of the bounding box annotations provided by [4]. CLEVR [3] images are collected by (1) sampling scene parameters such as object attributes, positions, lighting, and camera positions and (2) rendering images. However, the public CLEVR scene files do not contain the original bounding box coordinates and randomly jittered camera positions, which makes it impossible to precisely reconstruct the original bounding box coordinates

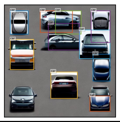
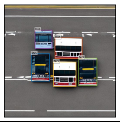
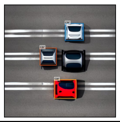
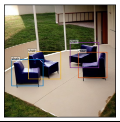
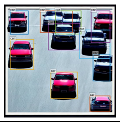
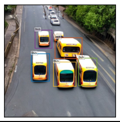
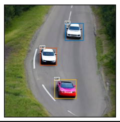
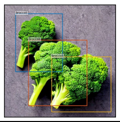
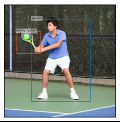
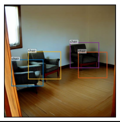
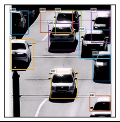
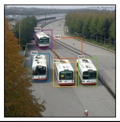
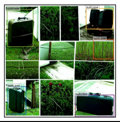
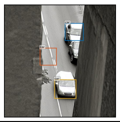
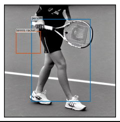
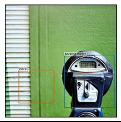
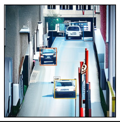
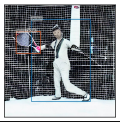
	Skill 1: Number		Skill 2: Position		Skill 3: Size		Skill 4: Combination	
	few	many	center	boundary	tiny	large	common	uncommon
Captions	4 chairs	10 cars	5 buses	5 suitcases	3 cars	3 broccolis	person is holding tennis racket	parking meter is next to clock
ControlNet								
GLIGEN								
ReCo								
ITERINPAINT								

Table 13. Example images generated by four different methods given four splits of caption and layouts from LAYOUTBENCH-COCO. For number/position/size skills, the captions have the prefix 'a photo of' before the '`[N] [objects]`' text.

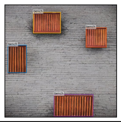
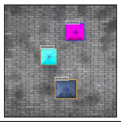
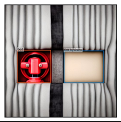
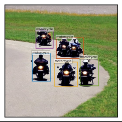
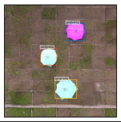
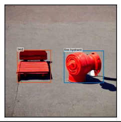
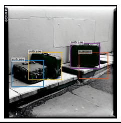
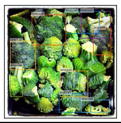
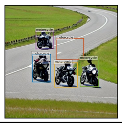
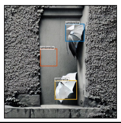
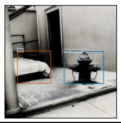
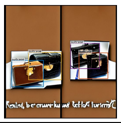
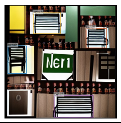
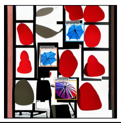
	Skill 1: Number		Skill 2: Position		Skill 3: Size		Skill 4: Combination	
	few	many	center	boundary	tiny	large	common	uncommon
Captions	4 suitcases	9 broccolis	5 motorcycles	4 benches	3 umbrellas	a surfboard	handbag is holding cell phone	fire hydrant is next to bed
ControlNet								
GLIGEN								
ReCo								
ITERINPAINT								

Table 14. Additional example images generated by four different methods given four splits of caption and layouts from LAYOUTBENCH-COCO. For number/position/size skills, the captions have the prefix 'a photo of' before the '`[N] [objects]`' text.

of CLEVR images.[3] Krishna *et al*. [4] provides bounding box annotations for CLEVR by approximating camera positions, but the annotations still have some errors. In Table 15, we compare (1) bounding box annotations provided by [4] (colored in blue) and (2) object detection results based on our LAYOUTBENCH-trained DETR (colored in red), where our DETR outputs boxes that bound the objects more tightly than the box annotations from [4]. On the test set of these re-rendered CLEVR images with precise bounding boxes, our DETR object detector could achieve 99% AP.

[3]https://github.com/facebookresearch/clevr-dataset-gen/blob/9742828c3667e81d5c381dbe1a0bcae4c1a7e89a/image_generation/render_images.py#L270-L273

D. Additional GAN Baseline

In addition to the denoising diffusion models (LDM and ReCo), we also experiment with Hintz *et al*. [2], a GAN-based layout-guided image generation approach as our baseline on CLEVR and LAYOUTBENCH layout experiments. As shown in Table 16, while Hintz *et al*. tend to

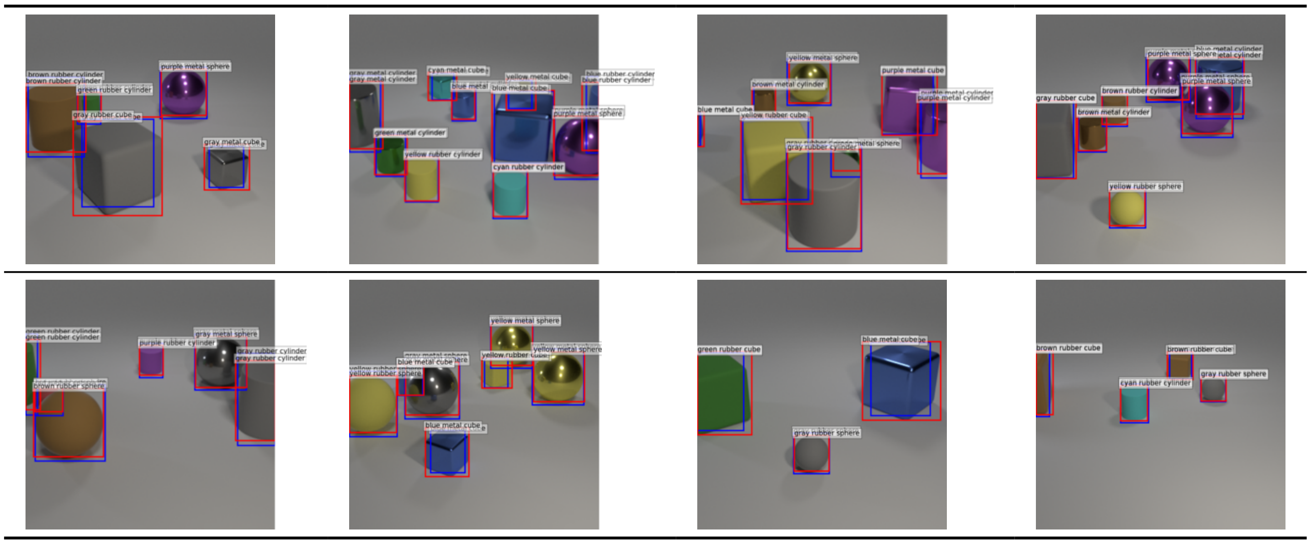


Table 15. CLEVR images with (1) bounding box annotations provided by Krishna *et al*. (2018) [4] (colored in blue) and (2) object detection results based on our LAYOUTBENCH-trained DETR (colored in red).

place objects in correct locations, the objects are much blurrier than the other diffusion models LDM, ReCo, and ITER-INPAINT. The low image quality makes the objects very hard to recognize (*i.e*., we could not tell whether a generated object is a cylinder, a cube, or a sphere), making the model achieve much worse image quality (*e.g*., 180.9 FID on CLEVR) and layout accuracy (*e.g*., 1.8 % AP on LAYOUTBENCH) metrics.

E. Additional Image Generation Samples on LAYOUTBENCH

In the following Table 17, Table 18, Table 19, Table 20, Table 21, Table 22, Table 23, and Table 24, we show additional image samples of LAYOUTBENCH and model generation results.

References

[1] Jaemin Cho, Jiasen Lu, Dustin Schwenk, Hannaneh Hajishirzi, and Aniruddha Kembhavi. X-LXMERT: Paint, Caption and Answer Questions with Multi-Modal Transformers. In *EMNLP*, 2020. 2

[2] Tobias Hinz, Stefan Heinrich, and Stefan Wermter. Generating multiple objects at spatially distinct locations. In *ICLR*, 2019. 5, 7

[3] 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*, pages 2901–2910, 2017. 4

[4] Ranjay Krishna, Ines Chami, Michael Bernstein, and Li Fei-Fei. Referring Relationships. In *CVPR*, 2018. 4, 5, 6

[5] Tsung Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *ECCV*, 2014. 2

[6] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-Shot Text-to-Image Generation. In *ICML*, 2021. 2

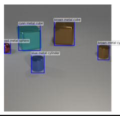
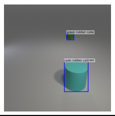
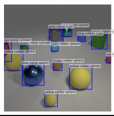
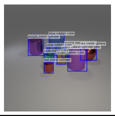
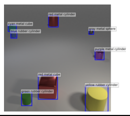
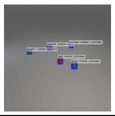
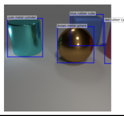
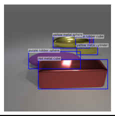
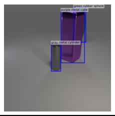
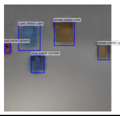
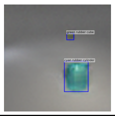
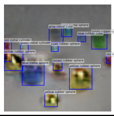
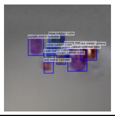
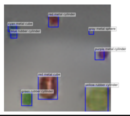
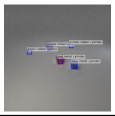
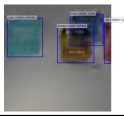
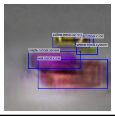
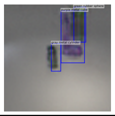
Method	CLEVR	LAYOUTBENCH							
	val	Number		Position		Size		Shape	
		few	many	center	boundary	tiny	large	horizontal	vertical
GT									
Hintz *et al*. [2]									

Table 16. Images generated by Hintz *et al*. [2] on CLEVR (ID) and LAYOUTBENCH (OOD) layouts. GT boxes are shown in blue.

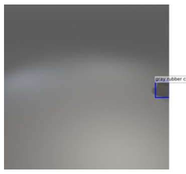
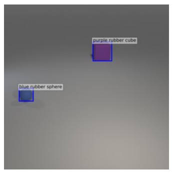
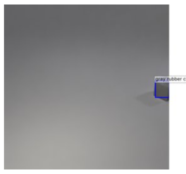
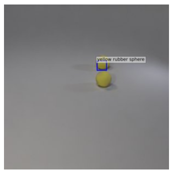
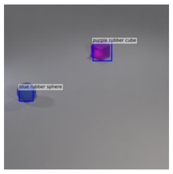
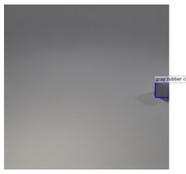
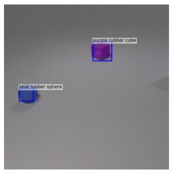
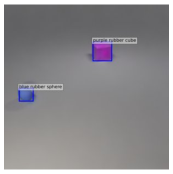
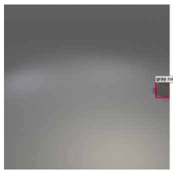
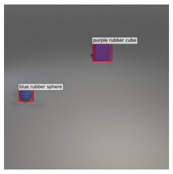
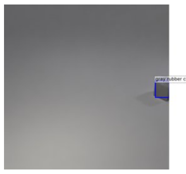
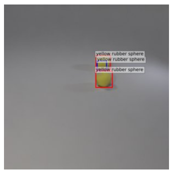
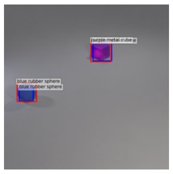
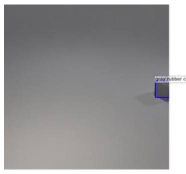
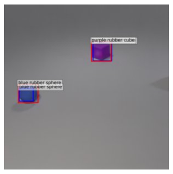
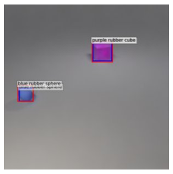
GT				
LDM				
ReCo				
ITERINPAINT (Ours)				
GT				
LDM				
ReCo				
ITERINPAINT (Ours)				

Table 17. Additional LAYOUTBENCH Number-few task samples. Top 4 rows: Images with GT boxes (in blue). Bottom 4 rows: Images with GT boxes (in blue) and object detection results (in red).

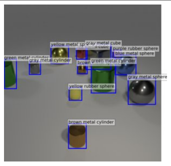
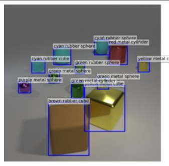
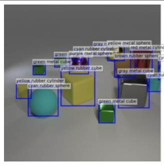
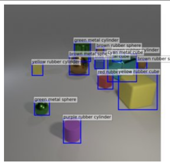
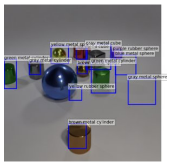
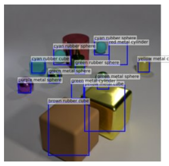
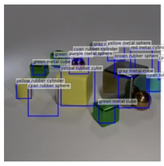
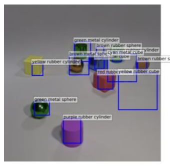
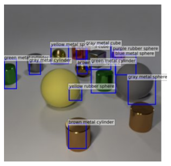
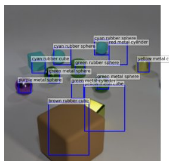
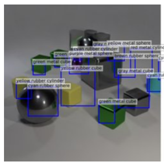
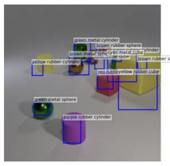
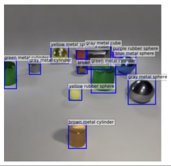
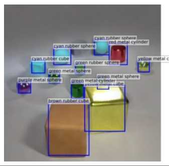
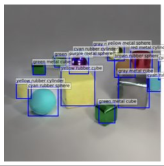
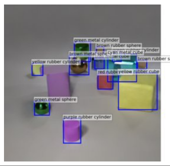
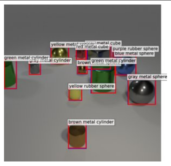
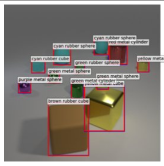
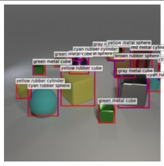
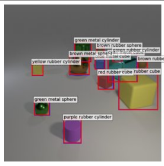
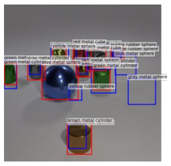
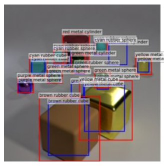
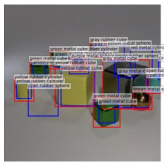
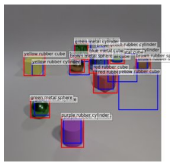
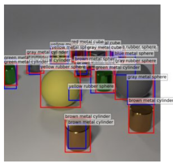
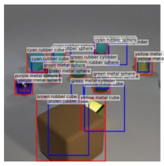
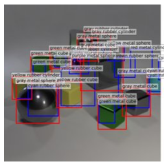
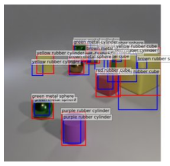
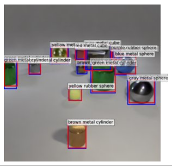
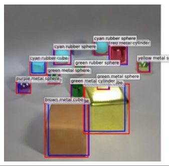
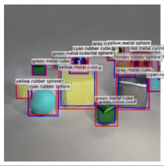
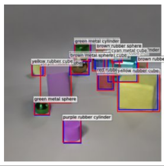
GT				
LDM				
ReCo				
ITERINPAINT (Ours)				
GT				
LDM				
ReCo				
ITERINPAINT (Ours)				

Table 18. Additional LAYOUTBENCH Number-many task samples. Top 4 rows: Images with GT boxes (in blue). Bottom 4 rows: Images with GT boxes (in blue) and object detection results (in red).

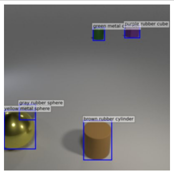
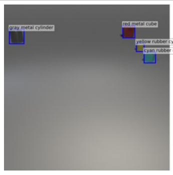
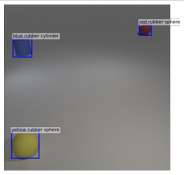
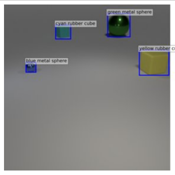
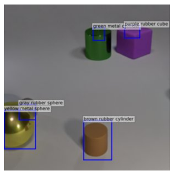
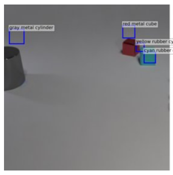
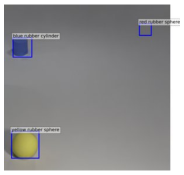
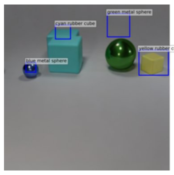
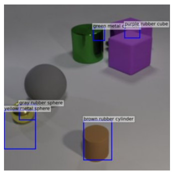
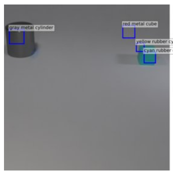
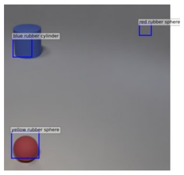
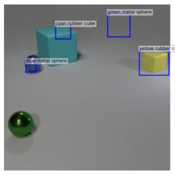
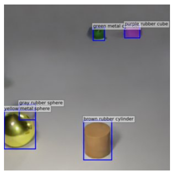
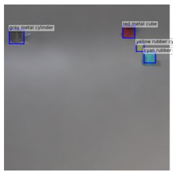
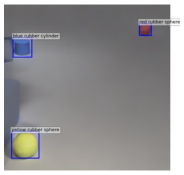
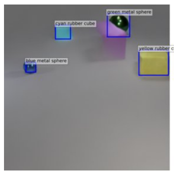
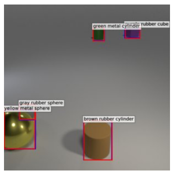
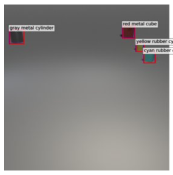
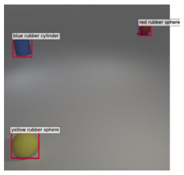
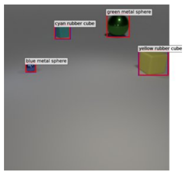
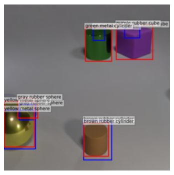
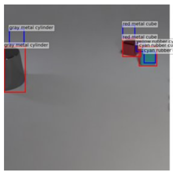
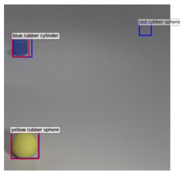
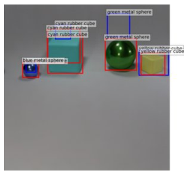
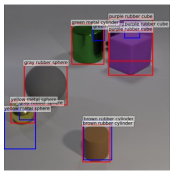
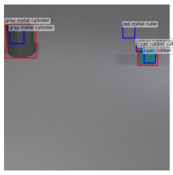
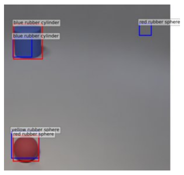
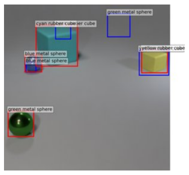
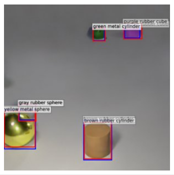
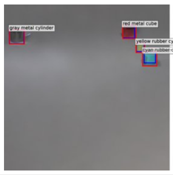
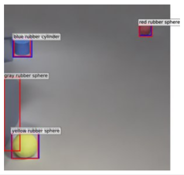
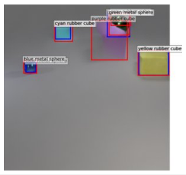
GT				
LDM				
ReCo				
ITERINPAINT (Ours)				
GT				
LDM				
ReCo				
ITERINPAINT (Ours)				

Table 19. Additional LAYOUTBENCH Position-boundary task samples. Top 4 rows: Images with GT boxes (in blue). Bottom 4 rows: Images with GT boxes (in blue) and object detection results (in red).

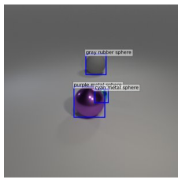
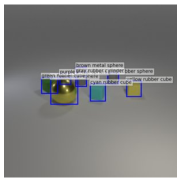
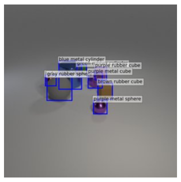
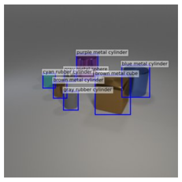
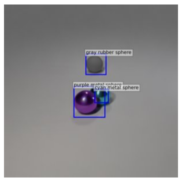
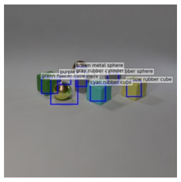
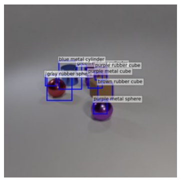
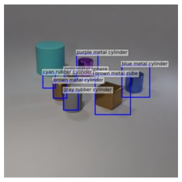
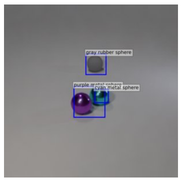
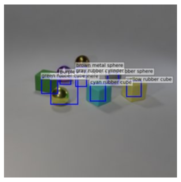
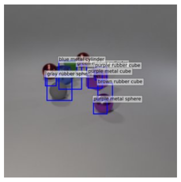
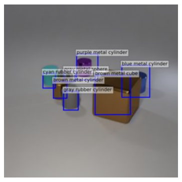
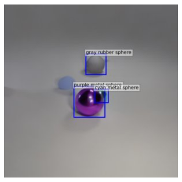
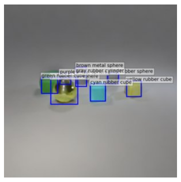
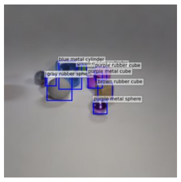
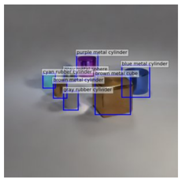
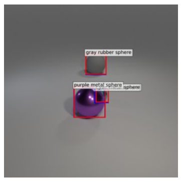
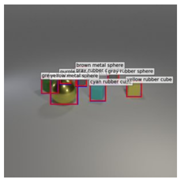
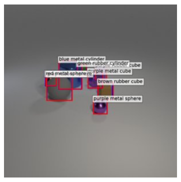
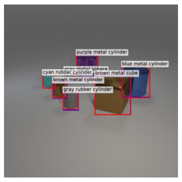
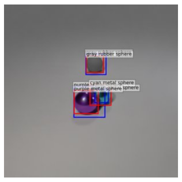
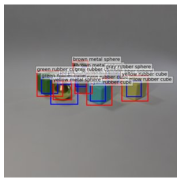
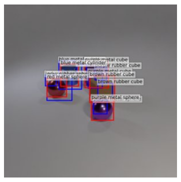
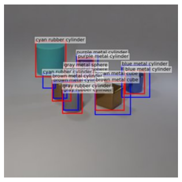
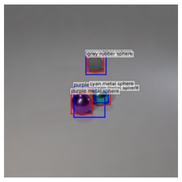
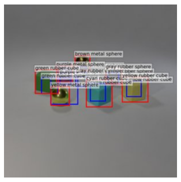
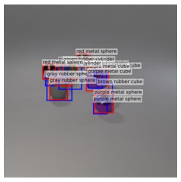
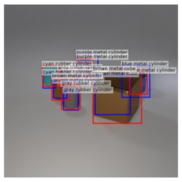
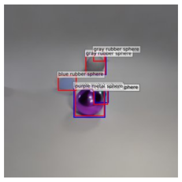
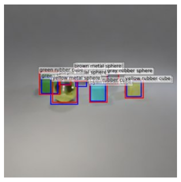
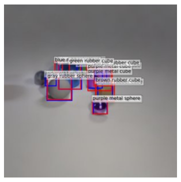
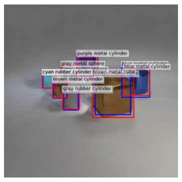
GT				
LDM				
ReCo				
ITERINPAINT (Ours)				
GT				
LDM				
ReCo				
ITERINPAINT (Ours)				

Table 20. Additional LAYOUTBENCH Position-center task samples. Top 4 rows: Images with GT boxes (in blue). Bottom 4 rows: Images with GT boxes (in blue) and object detection results (in red).

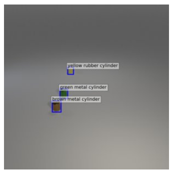
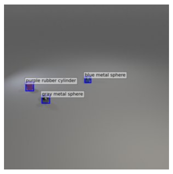
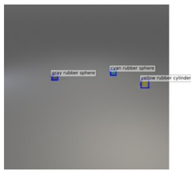
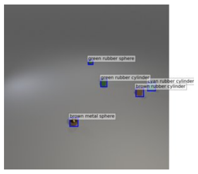
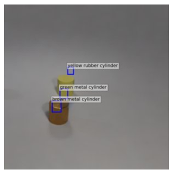
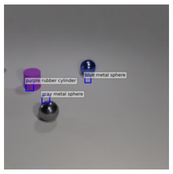
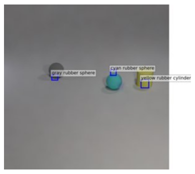
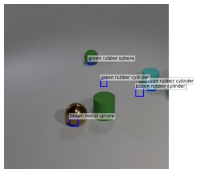
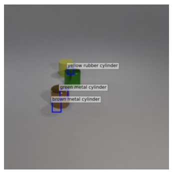
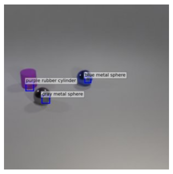
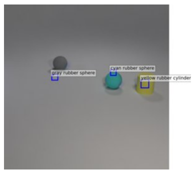
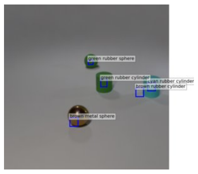
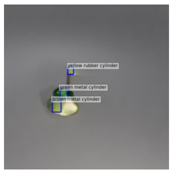
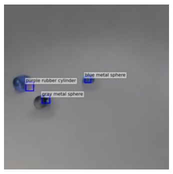
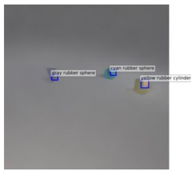
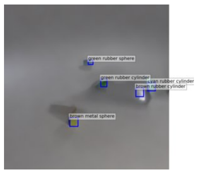
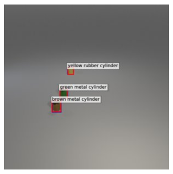
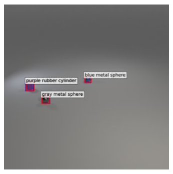
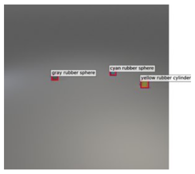
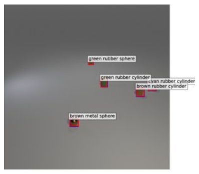
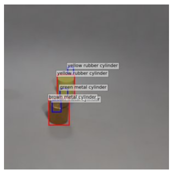
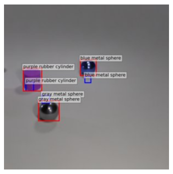
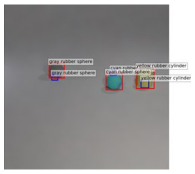
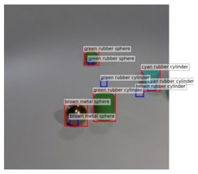
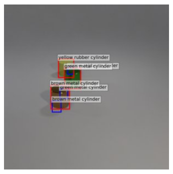
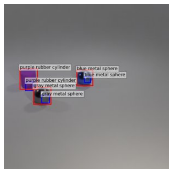
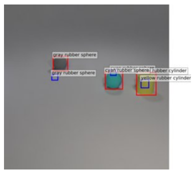
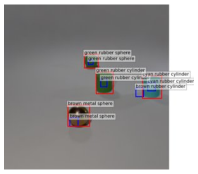
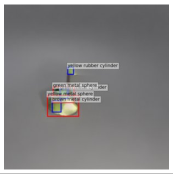
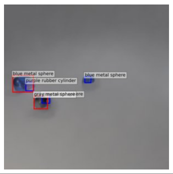
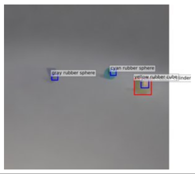
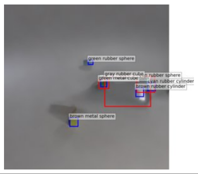
GT				
LDM				
ReCo				
ITERINPAINT (Ours)				
GT				
LDM				
ReCo				
ITERINPAINT (Ours)				

Table 21. Additional LAYOUTBENCH Size-tiny task samples. Top 4 rows: Images with GT boxes (in blue). Bottom 4 rows: Images with GT boxes (in blue) and object detection results (in red).

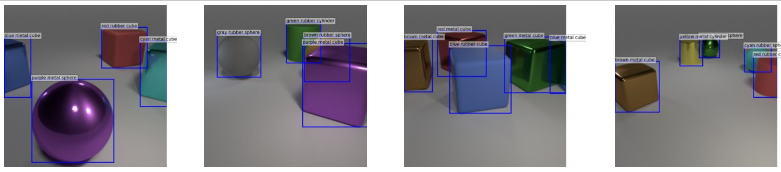
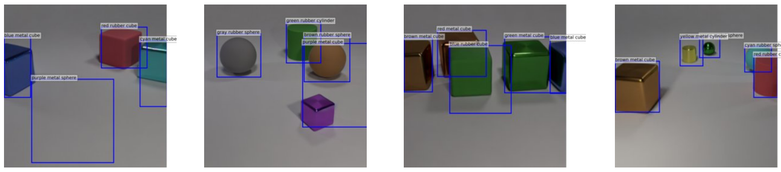
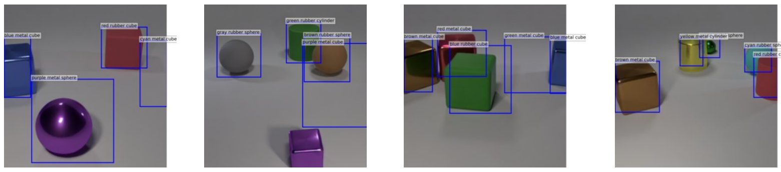
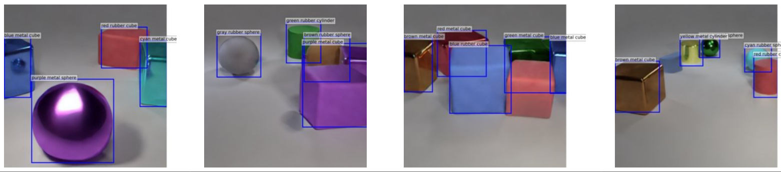
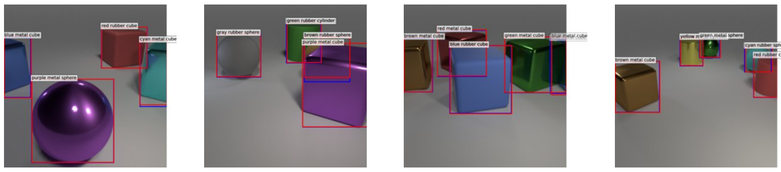
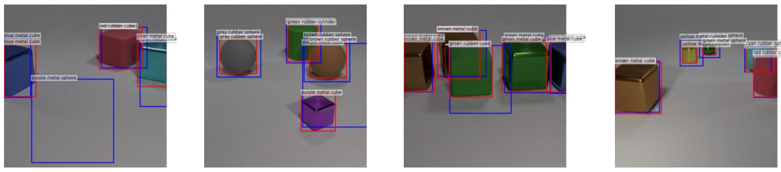
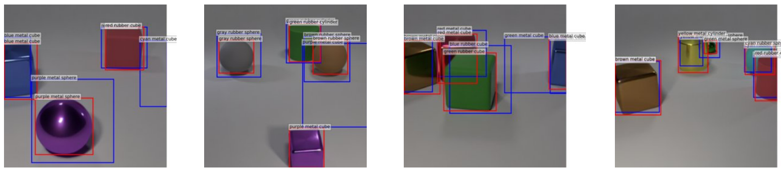
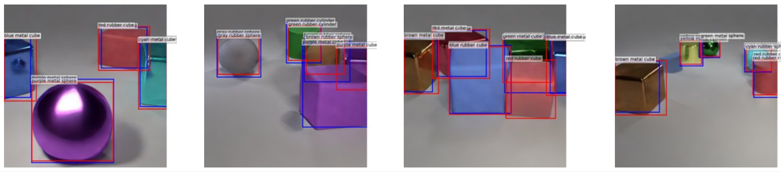
GT	
LDM	
ReCo	
ITERINPAINT (Ours)	
GT	
LDM	
ReCo	
ITERINPAINT (Ours)	

Table 22. Additional LAYOUTBENCH Size-large task samples. Top 4 rows: Images with GT boxes (in blue). Bottom 4 rows: Images with GT boxes (in blue) and object detection results (in red).

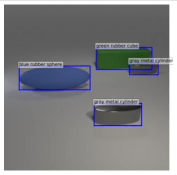
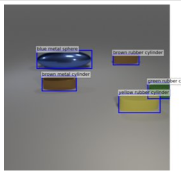
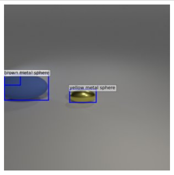
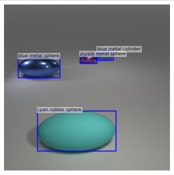
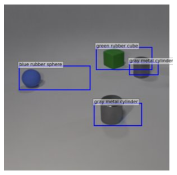
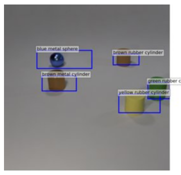
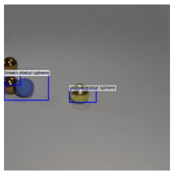
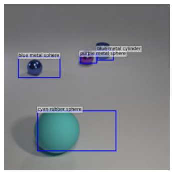
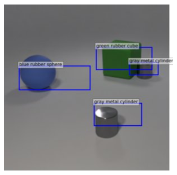
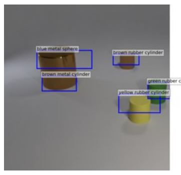
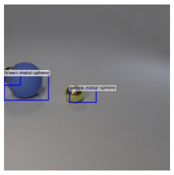
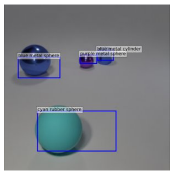
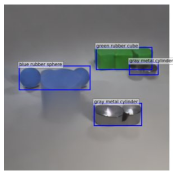
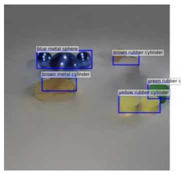
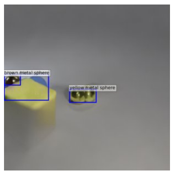
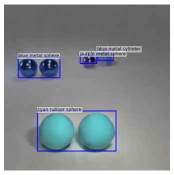
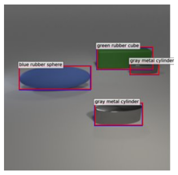
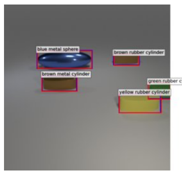
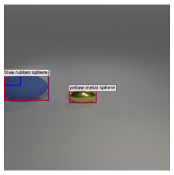
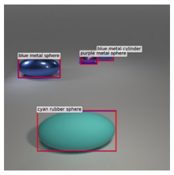
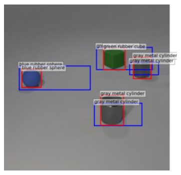
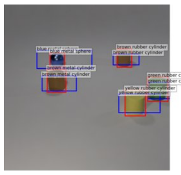
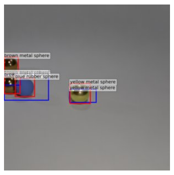
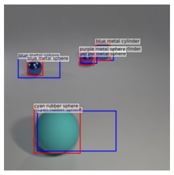
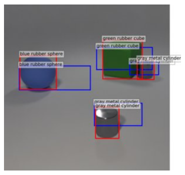
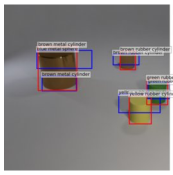
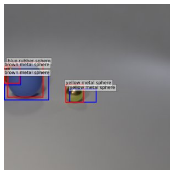
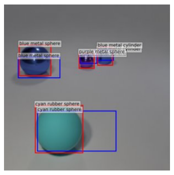
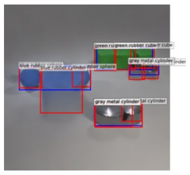
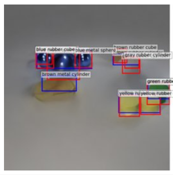
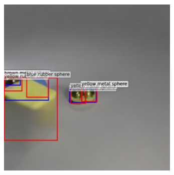
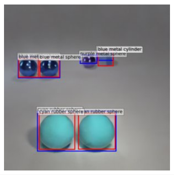
GT				
LDM				
ReCo				
ITERINPAINT (Ours)				
GT				
LDM				
ReCo				
ITERINPAINT (Ours)				

Table 23. Additional LAYOUTBENCH Shape-horizontal task samples. Top 4 rows: Images with GT boxes (in blue). Bottom 4 rows: Images with GT boxes (in blue) and object detection results (in red).

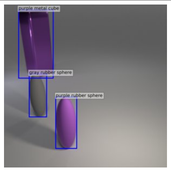
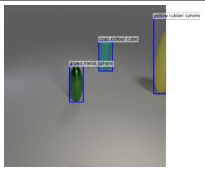
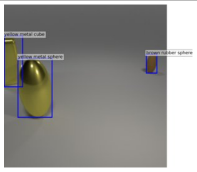
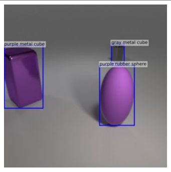
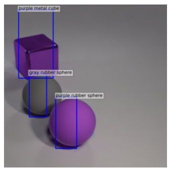
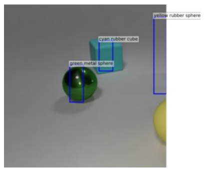
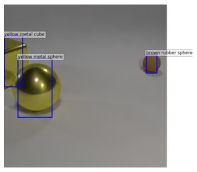
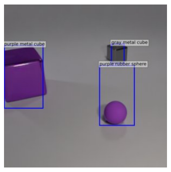
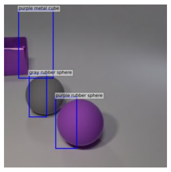
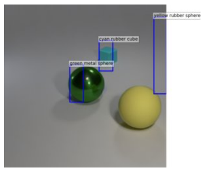
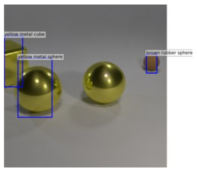
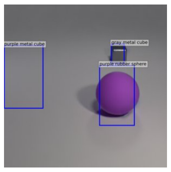
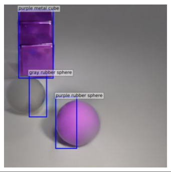
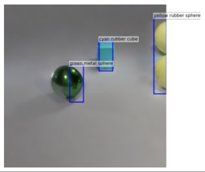
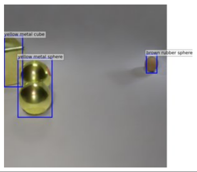
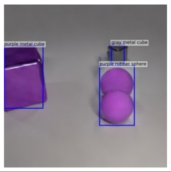
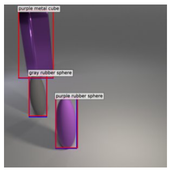
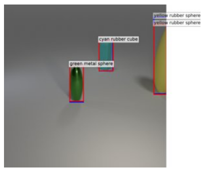
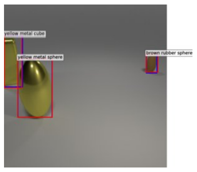
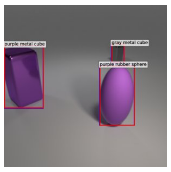
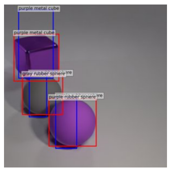
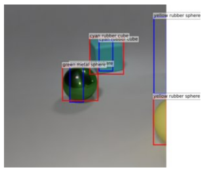
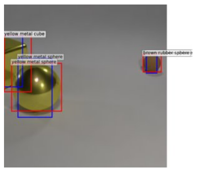
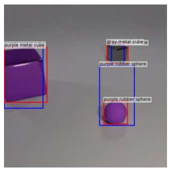
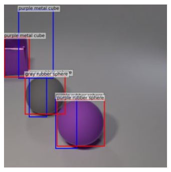
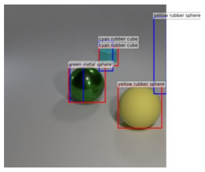
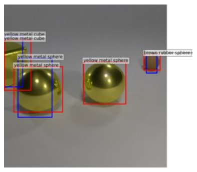
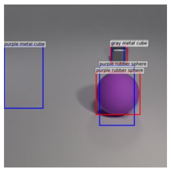
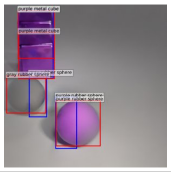
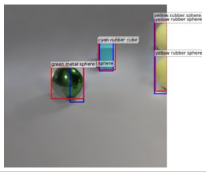
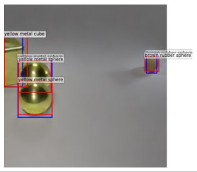
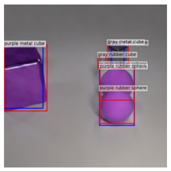
GT				
LDM				
ReCo				
IterInpaint (Ours)				
GT				
LDM				
ReCo				
IterInpaint (Ours)				

Table 24. Additional LayoutBench Shape-vertical task samples. Top 4 rows: Images with GT boxes (in blue). Bottom 4 rows: Images with GT boxes (in blue) and object detection results (in red).