Diagnostic Benchmark and Iterative Inpainting for Layout-Guided Image Generation

Jaemin Cho[1] Linjie Li[2] Zhengyuan Yang[2] Zhe Gan[2] Lijuan Wang[2] Mohit Bansal[1]
UNC Chapel Hill[1] Microsoft Research[2]
{jmincho, mbansal}@cs.unc.edu {lindsey.li, zhengyang, zhe.gan, lijuanw}@microsoft.com
https://layoutbench.github.io

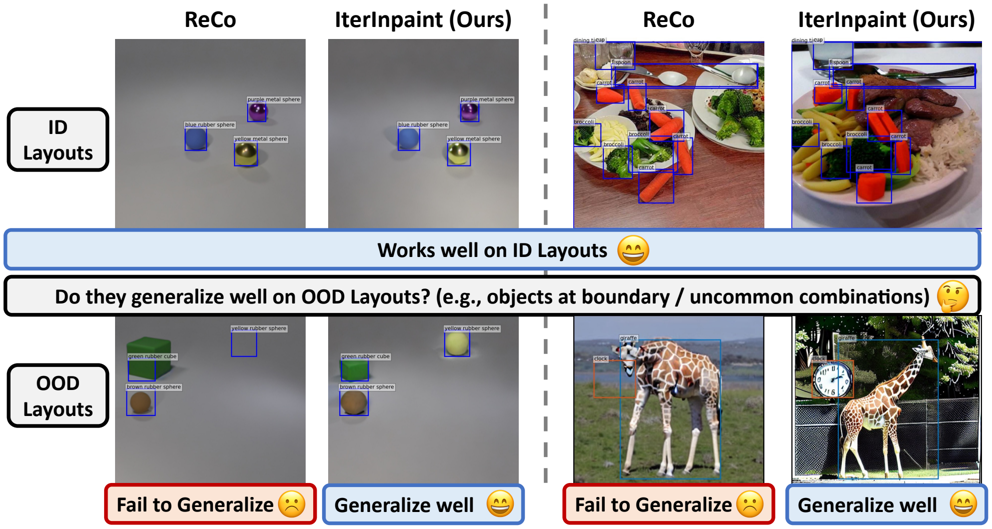


Figure 1. We propose LAYOUTBENCH (Sec. 3), a diagnostic benchmark for layout-guided image generation models with **out-of-distribution (OOD)** layouts in four skills: *number*, *position*, *size*, and *shape*. Existing models such as ReCo [46] fail on OOD layouts by misplacing objects. Next, we introduce ITERINPAINT (Sec. 4), a new baseline model with a better generalization on OOD layouts.

Abstract

Spatial control is a core capability in controllable image generation. Advancements in layout-guided image generation have shown promising results on in-distribution (ID) datasets with similar spatial configurations. However, it is unclear how these models perform when facing out-of-distribution (OOD) samples with arbitrary, unseen layouts. In this paper, we propose LAYOUTBENCH*, a diagnostic benchmark for layout-guided image generation that examines four categories of spatial control skills: number, position, size, and shape. We benchmark two recent representative layout-guided image generation methods and observe that the good ID layout control may not generalize well to arbitrary layouts in the wild (e.g., objects at the boundary). Next, we propose* ITERINPAINT*, a new baseline that generates foreground and background regions step-by-step via inpainting, demonstrating stronger generalizability than existing models on OOD layouts in* LAYOUTBENCH*. We perform quantitative and qualitative evaluation and fine-grained analysis on the four* LAYOUTBENCH *skills to pinpoint the weaknesses of existing models. We show comprehensive ablation studies on* ITERINPAINT*, including training task ratio, crop&paste vs. repaint, and generation order. Lastly, we evaluate the zero-shot performance of different pretrained layout-guided image generation models on* LAYOUTBENCH-COCO*, our new benchmark for OOD layouts with real objects, where our* ITERINPAINT *consistently outperforms SOTA baselines in all four splits.*

1. Introduction

With the advance of image generation systems that can synthesize diverse and realistic images, there is an increasing demand for controllable image generation systems that can precisely follow arbitrary spatial configurations defined by users. For this reason, recent work has focused on the task of layout-to-image generation [9, 11, 22, 24, 35, 40, 45, 52], which aims to generate images conditioned on multiple object bounding boxes and their paired object labels. Recent layout-guided text-to-image generation models [1, 23, 46] extend predefined object labels with open-ended regional captions, facilitating the models to generate open-set entities with the queried spatial configurations. With the recent advances in large-scale image generation models [12, 32, 35, 36, 47], newer layout-guided models [1, 23, 46] have shown promise in generating high-fidelity images following spatial configurations.

However, most experiments in these previous works are conducted in the in-distribution (ID) setting, where the queried spatial configuration shares a similar layout as the ones in the training samples. Hence, a natural question arises: how well do these image generation methods perform in real-world scenarios with arbitrary, unseen out-of-distribution (OOD) layouts (*e.g*., many more or larger/smaller or unusually positioned/shaped regions as compared to the training samples)? Recent studies [23, 46] use qualitative and human evaluations to interpret the model's generation capabilities on arbitrary spatial configurations. However, those studies focus on the method development and do not provide systematic benchmarks for spatial control in image generation. In this study, we aim to develop a benchmark to understand the status quo of image generation with arbitrary spatial configurations and further develop an iterative inpainting-based model to improve the OOD layout generalization.

To this end, we first propose **LAYOUTBENCH** (Sec. 3), a diagnostic benchmark featuring three properties as follows. **(1)** We define four categories for spatial control: number, position, size, and shape. LAYOUTBENCH systematically designs the out-of-distribution (OOD) testing queries for each skill, allowing easy comparison between different spatial configurations. **(2)** We evaluate images by layout accuracy in average precision (AP) to reflect the controllable generation quality. With the release of a well-performing category-balanced object detector that localizes generated objects, LAYOUTBENCH allows fair and easy comparison with prior works. **(3)** We choose to develop the benchmark based on the CLEVR simulator [20] to disentangle the factor of image generation quality from the interested spatial controllability. By simplifying the benchmark with simulated objects, LAYOUTBENCH can better reflect the true spatial control capabilities and avoid blindly favoring large-scale generation models that generate images with better visual qualities but do not understand spatial configurations.

Based on LAYOUTBENCH, we systematically evaluate two recent representative layout-guided image generation methods: LDM [35] and ReCo [46], where both models are initialized by the Stable Diffusion checkpoint. We perform quantitative and qualitative analyses and fine-grained split analyses on the four LAYOUTBENCH skills to pinpoint the weaknesses of different models. As depicted in Fig. 1, we find that both models fail on OOD layouts of LAYOUTBENCH, while they perform reasonably well on ID layouts.

Inspired by the OOD failures of existing models revealed by our LAYOUTBENCH benchmark, we next propose **ITERINPAINT**, a new baseline for layout-guided image generation (Sec. 4). Unlike existing methods that condition all the region configurations at a single generation step, ITERINPAINT decomposes image generation into multiple inpainting steps and iteratively updates each region at a time. By focusing on updating a single region at each time, the model can tackle unseen, complex layouts more robustly than existing methods. In experiments (Sec. 5), ITERINPAINT shows significantly better layout accuracy on OOD layouts and similar or better layout accuracy on ID layouts than prior works. We also provide comprehensive ablation studies on ITERINPAINT, including training task ratio, crop&paste *vs*. repaint-based update, and generation order. Lastly, we evaluate zero-shot performance of different pretrained layout-guided image generation models on LAYOUTBENCH-COCO, our new OOD layouts with real objects, where our ITERINPAINT outperforms other SOTA models in all four splits.

Our contributions are summarized as follows: **(1)** LAYOUTBENCH, a diagnostic benchmark for arbitrary spatial control capabilities of layout-guided image generation models in four criteria: number, position, size, and shape, where existing models often struggle (Sec. 3); **(2)** ITERINPAINT, a new baseline for layout-guided image generation that generates foreground and background in a step-by-step manner, which shows better generalization on OOD layout than prior works (Sec. 4); and **(3)** detailed qualitative/quantitative/sub-split evaluation of spatial control skills of different layout-guided image generation models, comprehensive ablation studies of ITERINPAINT design choices, and zero-shot evaluation of pretrained layout-guided image generation models on LAYOUTBENCH-COCO (Sec. 5).

2. Related Work

Text-to-Image Generation Models. In the text-to-image generation task, models generate images from natural language descriptions. Early deep learning-based models [28, 33, 44, 48] were based on Generative Adversarial Networks (GANs) [13]. Recently, multimodal language models and diffusion models have been widely used for this task. X-LXMERT [5] and DALL-E [31] introduce multimodal language models that take text as input and generate discrete image codes iteratively, where a vector-quantized autoencoder learns the mapping between image codes and pixels. LDM [35] and GLIDE [29] propose text-conditional diffusion models [18, 38] that iteratively update noisy images to clean images. Recent multimodal language models (*e.g*., Parti [47] and MUSE [4]) and diffusion models (*e.g*., Stable Diffusion [35], DALL-E 2 [32], and Imagen [36]) deliver high level of photorealism in zero-shot generation.

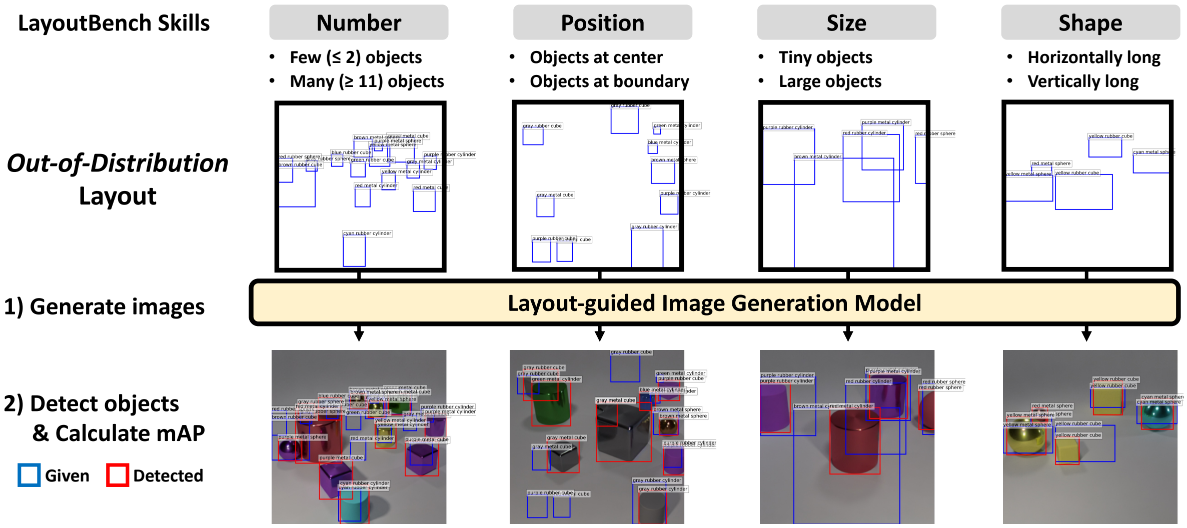


Figure 2. In LayoutBench, we measure 4 spatial control skills (number, position, size, shape) for layout-guided image generation. First, 1) we query the image generation models with OOD layouts. Then, 2) we detect the objects from the generated images, and calculate the layout accuracy in average precision (AP). In each image, the ground-truth boxes are shown in blue and the objects detected are shown in red. The images are generated by ReCo [46] trained on CLEVR [20], where it often misplaces (*i.e.*, many red boxes outside of blue boxes) or misses objects (*i.e.*, many blue boxes are missed) on OOD layouts from LayoutBench.

Layout-to-Image Generation Models. In the layout-to-image generation task, models generate images from layouts (*e.g.*, bounding boxes with paired text descriptions). Early models adopt GAN framework [39, 41, 51], where an adversarially trained convolutional generator is conditioned on layout input. Mirroring the success of the text-to-image models, recent layout-to-image models adopt multimodal language model (*e.g.*, Make-A-Scene [12]) and diffusion models (*e.g.*, LDM [35], ReCo [46], SpaText [1], GLIGEN [23], Universal Guided Diffusion [2]). While the prior works encode and decode all region inputs in a single step, IterInpaint decomposes image generation into multiple steps by focusing on generating one region at one time, showing better generalization on unseen OOD layouts.

Evaluation for Layout-to-Image Generation. The layout-to-image generation community has adopted two types of metrics used in text-to-image generation tasks: image quality and image-layout alignment. For image quality, Inception Score (IS) [37] and Fréchet Inception Distance (FID) [15] are commonly used. These metrics use a classifier pretrained on ImageNet [8] that mostly contains single-object images. Therefore, they are not well suited for evaluating images with more complex scenes [10]. To measure image-layout alignment, calculating FID on box crops (SceneFID) [41] and object classification accuracy on box crops [51] have been proposed. All these metrics summarize the performance of layout-to-image models in a single number, which does not reveal the skills in which the model is good versus the model is bad. In contrast to the existing metrics which do not pinpoint the model weakness, our LayoutBench measures four spatial layout control (number, position, size, and shape), to provide a more fine-grained analysis of region control capabilities.

3. LayoutBench

We introduce **LayoutBench**, a diagnostic benchmark for layout-guided image generation, with a focus on four spatial control skills. In the following, we discuss dataset (Sec. 3.1), layout accuracy (Sec. 3.2), and the poor generalizability of existing methods [35, 46] on LayoutBench, which motivates us to propose IterInpaint (Sec. 4).

3.1. Dataset

As illustrated in Fig. 2, LayoutBench evaluates spatial control capability in 4 skills (number, position, size, shape), where each skill consists of 2 different OOD layout splits, *i.e.*, in total 8 tasks = 4 skills × 2 splits. To disentangle the spatial control from other aspects in image generation, such as generating diverse objects, LayoutBench keeps the same object configurations as CLEVR [19], whose objects have 3 shapes, 2 materials, and 8 colors (48 combinations in total). Images in LayoutBench are collected in two steps: (1) sample scenes for each skill, where a scene is defined by the objects and their positions, (2) render images from the scenes with Blender [7] simulator and obtain bounding box layouts. In total, we collect 8K images for LayoutBench evaluation, with 1K images per task. In Tab. 1, we show example images with ID and OOD layouts. We explain the scene configurations below.

In-distribution: CLEVR. All scenes have 3∼10 objects. These objects are positioned evenly on the canvas, with-

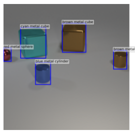
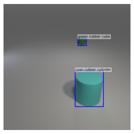
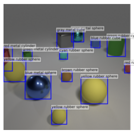
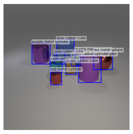
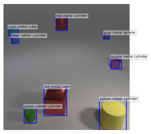
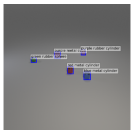
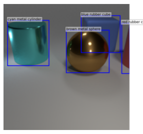
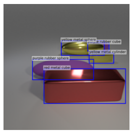
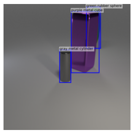
ID layout	OOD layout (LayoutBench)							
CLEVR	Skill 1: Number		Skill 2: Position		Skill 3: Size		Skill 4: Shape	
	few	many	center	boundary	tiny	large	horizontal	vertical
								

Table 1. Example images with ID (CLEVR) and OOD (LayoutBench) layouts. GT boxes are shown in blue.

out much occlusion between them. In terms of size, the rendered objects are in one of two scales $\{3.5, 7\}$. For shape, the bounding box for each object is an almost perfect square. For each of the skills below, we only alter the configurations specific to that skill, while keeping the remaining configurations the same as CLEVR.

Skill 1: Number. This skill involves generating images with a specified number of objects. In contrast to the ID CLEVR images with 3∼10 objects, we evaluate models on two OOD splits: (1) **few**: images with 0∼2 objects; (2) **many**: images with 11∼16 objects.

Skill 2: Position. This skill involves generating images with objects placed at specific positions. Different from ID CLEVR images featuring evenly distributed object position without much occlusion between objects, we design two OOD splits: (1) **center**: objects are placed at the center, thus leading to more occlusions; (2) **boundary**: objects are only placed on boundaries (top/bottom/left/right).

Skill 3: Size. This skill involves generating images with objects of a specified size. We construct two OOD splits: (1) **tiny**: objects with scale 2; (2) **large**: objects with scale $\{9, 11, 13, 15\}$. In comparison, the objects in CLEVR images have only two scales $\{3.5, 7\}$. We use 3∼5 objects for this skill, as we find that using more than this number of large objects can often obstruct the object visibilities.

Skill 4: Shape. This skill involves generating images with objects of a specified aspect ratio. As the objects in CLEVR images mostly have square aspect ratios, we evaluate models with two OOD splits: (1) **horizontal**: objects in which one of the horizontal (x/y) axes are 2 or 3 times longer than the other axis, leading to object bounding boxes with an aspect ratio (width:height) of 2:1 or 3:1; (2) **vertical**: objects whose vertical (z) axis are 2 or 3 times longer than horizontal (x/y) axes, resulting in object bounding boxes with an aspect ratio of 1:2 or 1:3. We use 3∼5 objects for this skill, as we find that using more than this number of objects can often obstruct the object visibilities.

3.2. Layout Accuracy Evaluation

As illustrated in Fig. 2, we evaluate models with four spatial control skills: number, position, size, and shape. Since existing metrics FID and SceneFID measure how the overall distribution of Inception v3 [42] mean-pool features of generated images/patches is similar to the feature distribution of ground-truth images/patches, they are less effective in measuring how accurately each generated image follows the input layout [10]. Following previous analyses [6, 16], we evaluate the skills based on how well an object detector can detect the object described in the input layout. To better capture the objects with uncommon sizes, positions, and aspect ratios etc, we train DETR [3] on separately generated 5K training images for each of 8 tasks, with 40K total images. We initialize DETR parameters from the official checkpoint with ResNet101 [14] backbone pretrained on the COCO [25] train 2017 split. Following object detection literature [3, 34, 53], we report average precision (AP).

We evaluate two recent layout-guided image generation models: LDM [35] and ReCo [46], trained on CLEVR. As shown in Fig. 1, they fail on LayoutBench by ignoring objects, misplacing objects, or placing wrong objects. We closely examine the experiment results in Sec. 5.

4. IterInpaint

To improve the generalizability of OOD layouts, we propose IterInpaint, a new layout-guided image generation method based on **iter**ative **inpaint**ing. Unlike previous methods [35, 46] that generate all objects simultaneously in a single step, IterInpaint decomposes the image generation process into multiple steps and uses a text-guided inpainting model to update foreground and background regions step-by-step. In what follows, we briefly recap Stable Diffusion, which we build IterInpaint on (Sec. 4.1), describe how we extend the Stable Diffusion for layout-guided inpainting (Sec. 4.2), and introduce iterative foreground/background inpainting (Sec. 4.3).

4.1. Preliminaries: Stable Diffusion

We implement the IterInpaint model by extending Stable Diffusion, a public text-to-image model based on Latent Diffusion [35]. Stable Diffusion consists of (1) a CLIP ViT-L/14 [30] text encoder $\texttt{CLIP}_{text}(s)$ that encodes a prompt s into a 512-dimensional vector, (2) an autoencoder $(E(x), D(z))$ with downsampling factor of 8, which embeds an image x into a 4-dimensional latent space $z_0 \in$

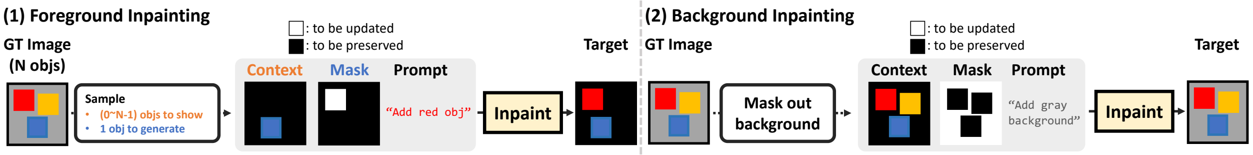
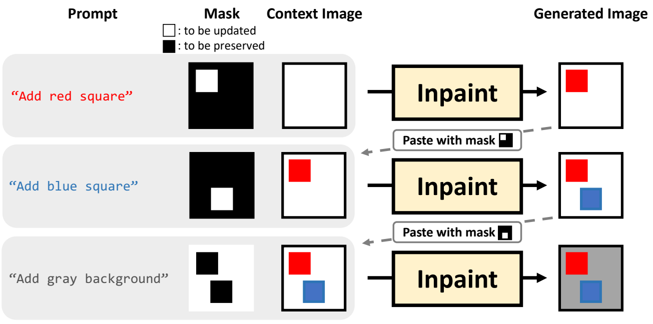


Figure 3. ITERINPAINT **Training.** Our model is trained with (1) foreground and (2) background inpainting tasks (Sec. 4.3).



```
def iterinpaint(caps, boxes):
    # Empty image
    img = Image.new((W,H))
    # Add foreground
    for cap, box in zip(caps, boxes):
        prompt = f"Add {cap}"
        mask = get_obj_mask(box)
        gen = inpaint(img, prompt, mask)
        img = (1-mask) * img + mask * gen
    # Add background
    prompt = "Add gray background"
    mask = get_bg_mask(boxes)
    gen = inpaint(img, prompt, mask)
    return gen
```

Figure 4. ITERINPAINT **Inference.** Illustration (left) and Python pseudocode (right) of layout-guided image generation with ITERINPAINT (Sec. 4.3). At each iteration, the inpainting model takes the prompt, mask, and previous image as inputs and generates a new image.

$\mathcal{R}^{(4,H,W)}$, and (3) a diffusion U-Net ϵ_θ that performs denoising steps in the latent space given timestamp t and CLIP text encoding (conditioned via cross-attention). The model is trained with the following objective: $L_{LDM} = \mathbb{E}_{s,z_0,t,\epsilon}[||\epsilon - \epsilon_\theta(z_t, t, \texttt{CLIP}_{text}(s))||_2^2]$.

4.2. Extending Stable Diffusion for Inpainting

Our ITERINPAINT method decomposes complex scene generation into multiple steps, where each step is a text-guided inpainting [50] process. Concretely, a model completes an image region, given a context image, a binary mask indicating the region, and a text description of the region. To enable inpainting, we extend the U-Net of Stable Diffusion to take the mask and a context image as additional inputs. We use a binary mask of the same size as the image, indicating the region to be updated (1: to be updated; 0: to be preserved). To encode the mask and context image, we add 5 additional channels to the U-Net's first convolutional layer, where the first dimension is used to encode the spatially downsampled mask $m \in [0,1]^{(H,W)}$, and the remaining 4 dimensions are for encoding the latent vector of the context image $z_0^{ctx} = E(x^{ctx}) \in \mathcal{R}^{(4,H,W)}$. The resulting layout-guided inpainting model takes a context image x^{ctx}, a text prompt s, and a binary mask m, as input and generates an image $x^{gen} = \texttt{inpaint}(x^{ctx}, s, m)$. Next, we describe the training and inference process of iterative inpainting.

4.3. Iterative Inpainting

ITERINPAINT decomposes the image generation process into two phases: (1) step-by-step generation of each bounding box/mask (foreground), and (2) filling the rest of the images (background). This decomposition would make each generation step easier by allowing the model to focus on generating a single foreground object or background.

During training, as shown in Fig. 3, we use a single objective to cover both foreground/background inpainting by giving the model a different context image and mask: (1) foreground inpainting - we sample context objects (from N GT objects) to show, then sample an object to generate; (2) background inpainting - we mask out all objects, and generate the background. We explore different ratios to sample the two training tasks in Sec. 5.4, and find a 30% and 70% ratio for foreground and background inpainting tasks gives the best performance. We train our model with the modified latent diffusion objective [35], $L_{IterInpaint} = \mathbb{E}_{s,z_0,t,\epsilon}[||\epsilon - \epsilon_\theta(z_t, t, \texttt{CLIP}_{text}(s), m, z_0^{ctx})||_2^2]$, where U-Net is additionally conditioned on the mask m and the previous image z_0^{ctx}.

During inference, as shown in Fig. 4, we iteratively update foreground objects and background, starting from a blank image. For each step, we update the image by composing context image x^{ctx} and the generated image x^{gen} using a mask m: $x^{new} = (1-m) * x^{gen} + m * x^{ctx}$. For a layout with N objects, our ITERINPAINT method generates the final image with $N+1$ (foreground + background) iterations. Overall, ITERINPAINT allows users to control the generation order of each region and interactively manipulate the image from an intermediate generation step.

5. Experiments and Analysis

5.1. Experimental Setup

In addition to our ITERINPAINT, we evaluate two recent and strong layout-guided image generation models, LDM [35], and ReCo [46]. To focus on layout control evaluation, we match the implementation details of three models.

Method	CLEVR	LayoutBench								
		Number		Position		Size		Shape		
	val	few	many	center	boundary	tiny	large	horizontal	vertical	Avg
GT (Oracle)	60.5/93.5	94.3/99.7	92.0/99.0	90.9/90.9	90.8/99.4	82.4/100.0	96.6/99.4	89.7/99.0	89.0/98.4	90.7/98.2
GT shuffled	0.0/0.0	0.1/0.1	0.0/0.0	0.0/0.0	0.0/0.0	0.0/0.0	0.0/0.0	0.0/0.0	0.0/0.0	0.0/0.0
LDM	54.5/**91.8**	14.0/48.7	4.7/20.7	5.5/28.0	5.9/15.1	0.0/0.0	37.8/68.2	2.0/12.8	9.3/38.5	9.9/29.0
ReCo	44.0/89.0	8.5/36.9	2.5/12.7	2.8/17.4	2.5/8.7	0.0/0.0	32.4/70.5	3.0/**19.3**	8.7/37.8	7.6/25.4
IterInpaint (Ours)	**57.2**/90.8	**50.4/80.5**	**52.4/87.7**	**49.6/83.8**	**50.1/82.1**	**2.4/7.9**	**63.1/92.6**	**4.7**/18.5	**19.3/60.1**	**36.5/64.1**

Table 2. Layout accuracy in AP/AP$_{50}$ (%) on CLEVR and LayoutBench. Best (highest) values are **bolded**.

Method	CLEVR	LayoutBench
LDM	3.4/**13.0**	31.1/57.9
ReCo	**2.8**/13.6	**30.4**/58.2
IterInpaint (Ours)	12.7/36.3	31.4/**49.0**

Table 3. Image quality in FID/SceneFID on CLEVR and LayoutBench. Best (lowest) values are in **bolded**.

Dataset details. We train models on the 70K training images in CLEVR [20]. As the original CLEVR dataset does not provide the bounding box annotations, we use the bounding box annotations provided by [21]. The original images have 480x360 (WxH) sizes. For training, we resize the images into 768x512 and center crop to 512x512.

Model details. We initialize all model parameters with Stable Diffusion v1 checkpoints. We train all models for 20K steps with batch size 128 (*i.e.*, single batch at each of 16 V100 GPUs with 8 gradient accumulation steps), and AdamW optimizer [27] with constant learning rate 1e-4. Following [46], we update U-Net and CLIP text encoder parameters, while freezing the autoencoder. During inference, we use classifier-free guidance [17] scale of 4.0 and 50 PLMS [26] steps.

Bounding box encoding. For LDM and ReCo, we quantize each of the bounding box coordinates (x_1, y_1, x_2, y_2) into 1000 quantized bins. For LDM, we learn 48 class embeddings for CLEVR objects. We describe the layout by concatenating the list of object class tokens and quantized bounding boxes (*e.g.*, "`<020> <230> <492> <478> <cls23> <121>` $\cdots$") and encode it with CLIP text encoder. Unlike LDM, ReCo takes the text description for each region instead of class embedding (*e.g.*, "`<020> <230> <492> <478> cyan metal sphere <121>` $\cdots$") as input.

Evaluation metrics. For quantitative evaluation, we measure layout accuracy and image quality. Layout accuracy is measured by AP (average precision) based on DETR-R101-DC5 [3], as mentioned in Sec. 3.2. Higher AP indicates that the generated images follow the input layouts more closely. FID [15] and SceneFID [41] are adopted to measure image quality. Lower FID (SceneFID) indicates that the generated images (boxes) have a more similar feature distribution to the ground-truth ones.

5.2. Evaluation on LayoutBench

Quantitative evaluation. We first evaluate the layout accuracy on generated images in Tab. 2. The first row shows the layout accuracy based on the ground-truth (GT) images. Our object detector evaluator can achieve high accuracy on both CLEVR and LayoutBench datasets, showing the high reliability of the detection-based layout accuracy evaluation results. Especially on LayoutBench, our detector achieves above 98% AP$_{50}$.[1] The second row (GT shuffled) shows a setting where given a target layout, we randomly sample an image from the GT images to be the generated image. The 0% AP on both CLEVR and LayoutBench means that it is impossible to obtain high AP by only generating high-fidelity images but in the wrong layouts.

In the bottom half of Tab. 2, we see that while all 3 models achieve high layout accuracy with above 89% AP$_{50}$ on CLEVR, the layout accuracy drop by large margins on LayoutBench, showing the ID-OOD layout gap. Specifically, LDM and ReCo fail substantially on LayoutBench across all skill splits, with an average performance drop of 57$\sim$70% per skill on AP$_{50}$, compared to the high AP on in-domain CLEVR validation split. Both models especially struggle with layout configurations with *many*, *tiny*, *horizontally-shaped* objects and when objects are placed in *center/boundary*. This is not surprising, as we have shown in Fig. 1; we will also show qualitative evaluation later in Tab. 4 that LDM and ReCo demonstrate poor generalizability to OOD layouts in LayoutBench.

In contrast, IterInpaint can generalize better to OOD layouts in LayoutBench while maintaining or even slightly improving the layout accuracy on ID layouts in CLEVR. Specifically, we observe an average performance gain of 49.5% for Number, 61.3% for Position, 15.0% for Size, and 10.7% for Shape in AP$_{50}$, compared to LDM and ReCo. Even on the extremely challenging Size-tiny split, where LDM and ReCo fails to render any tiny objects onto the given positions, IterInpaint can at least break the zero performance. Another challenging case is the shape-

[1] The AP of GT CLEVR images (60.5) is a bit lower than that of GT LayoutBench (90.7), because CLEVR bounding box annotations provided by Krishna *et al.* (2018) [21] have minor errors. On our re-rendered CLEVR images with precise bounding boxes, the object detector could achieve 99% AP (see appendix for details).

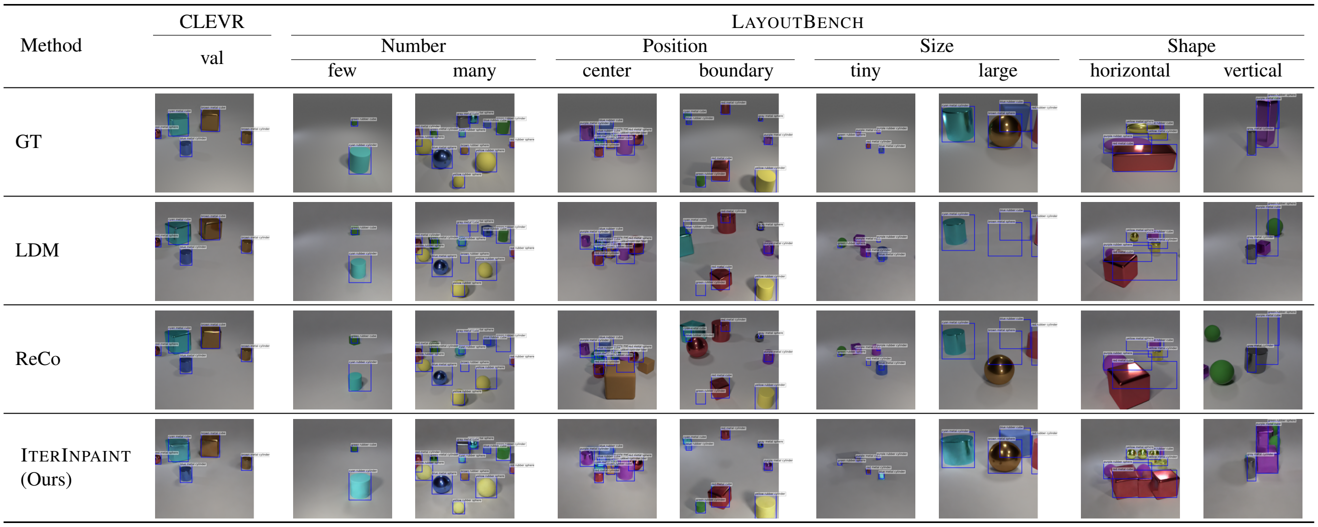


Table 4. Comparison of generated images on CLEVR (ID) and LayoutBench (OOD). GT boxes are shown in blue.

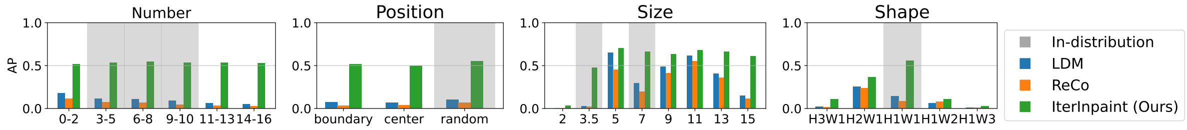


Figure 5. Detailed layout accuracy analysis with fine-grained splits of 4 LayoutBench skills. In-distribution (same attributes to CLEVR) splits are colored in gray. For the Shape skill, the splits are named after their height/width ratio (*e.g.* H2W1 split consists of the objects with a 2:1 ratio of height:width).

horizontal split, where all three models struggle, we further conduct detailed analysis on the difficulty levels of each split and the pain points of each model in Sec. 5.3.

In Tab. 3, we compare the quality of generated images by reporting the FID/SceneFID scores. On CLEVR, the LDM and ReCo achieves better FID/SceneFID metrics than IterInpaint, indicating that the strong layout control performance of IterInpaint comes with a trade-off in these image quality metrics. However, on LayoutBench, the three models achieve similar FID scores, despite the significant layout errors of LDM and ReCo, which suggests that image quality measures alone are not sufficient for evaluating layout-guided image generation [10] and further justify using layout accuracy to examine layout control closely.

Qualitative evaluation. Tab. 4 compares the GT images and the images generated by the three models. On CLEVR, all three models can follow the ID layout inputs to place the correct objects precisely. On LayoutBench, LDM and ReCo often make mistakes, such as generating objects that are much smaller (*e.g.*, Number-few) / bigger (*e.g.*, Size-tiny, Position-center) than the given bounding boxes and missing some objects (*e.g.*, Number-many, Position-center, Position-boundary, Size-large). However, IterInpaint can generate objects more accurately aligned to the given bounding boxes in general, consistent with the higher layout accuracy in Tab. 2. Especially for the extremely small bounding boxes in Size-tiny, only IterInpaint, among the three models, generates objects that fit. Interestingly, on Shape-horizontal/Shape-vertical, while all three struggle to generate long objects that are not seen in CLEVR, IterInpaint tries to fill the given long bounding boxes by generating multiple objects. More qualitative examples per skill are included in Appendix.

5.3. Fine-grained Skill Analysis

We perform a more detailed analysis on each LayoutBench skill to better understand the challenges presented in LayoutBench and examine each method's weakness. Specifically, we divide the 4 skills into more fine-grained splits to cover both in-distribution (ID; CLEVR configurations) and out-of-distribution (OOD; LayoutBench configurations) examples. We sample 200 images for each split and report layout accuracy in Fig. 5.

Overall. Comparing across 4 skills, the majority of Size skill splits (except for size=2) are the least challenging, while the Position/Number skill is the most challenging. IterInpaint significantly outperforms LDM and ReCo on all splits. Among the other two, LDM has slightly higher scores than ReCo overall.

Number. As the number of objects increases in the first plot of Fig. 5, LDM and ReCo performance decreases, while the IterInpaint performance remains consistent.

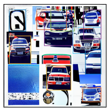
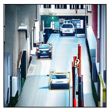
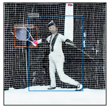
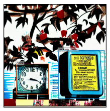
Skill 1: Number		Skill 2: Position		Skill 3: Size		Skill 4: Combination	
few	many	center	boundary	tiny	large	common	uncommon
4 chairs	10 cars	5 buses	5 suitcases	3 cars	3 broccolis	person is holding tennis racket	parking meter is next to clock
							

Table 5. Example images generated by IterInpaint given four splits of caption and layouts from LayoutBench-COCO.

Method	Number	Position	Size	Combination
ControlNet	9.2	15.3	10.8	6.4
GLIGEN	30.7	36.4	33.3	36.3
ReCo	30.9	38.9	24.1	18.7
IterInpaint (Ours)	**31.4**	**39.1**	**33.5**	**44.1**

Table 6. Zero-shot Layout Accuracy in AP (%) on LayoutBench-COCO.

Position. As shown in the second plot of Fig. 5, there is a slight ID-OOD performance gap for all three models. The models perform similarly on boundary and center splits, while slightly lower than the random ID split.

Size. As shown in the third plot of Fig. 5, the models are better at generating large objects than small objects. Notably, all models fail at size=2, the smallest object scale in our experiment. As shown in Tab. 4's Size-tiny column, LDM and ReCo tend to generate bigger objects from small bounding boxes, whereas IterInpaint could correctly generate small objects in the right location but misses the details of right shapes or attributes.

Shape. As shown in the last plot of Fig. 5, there is a strong ID-OOD gap for all three models. The models generate vertically long (H2W1 and H3W1) better than horizontally long (H1W2 and H1W3) objects. From our manual analysis, there were some trends for models to prioritize fitting the height to the width of the bounding boxes. This results in trends of generating small square boxes for horizontally long boxes that are too small to cover the box and generating big square boxes that can sometimes cover some vertically long boxes (see appendix for more examples).

5.4. Ablation of IterInpaint

We conduct ablation studies of IterInpaint design choices: (1) Pasting (default) *vs*. Repaint based update, (2) training task ratio for foreground & background inpainting, and (3) object generation order. In summary, we found that (1) repaint-based update suffers from error propagation, (2) the 3:7 fg/bg training task ratio performs best, but other task ratios perform similarly, and (3) IterInpaint is robust in arbitrary generation orders, allowing flexible object layout manipulation without full image re-rendering. Please see appendix for detailed analysis.

5.5. LayoutBench-COCO: Zero-shot Evaluation of Layout-guided Image Generation Models

Although our main focus is to provide a benchmark of layout-guided image generation models with full control, including the same computation and training data with arbitrary objects (*e.g*., blue metal cube), we also test the spatial control capabilities of existing pretained models with layouts of real objects (*e.g*., cars) in zero-shot. For this, we create LayoutBench-COCO, a real object version of LayoutBench with 4 splits (Number, Position, Size, Combination), whose objects are from MS COCO [25]. The new 'combination' split consists of layouts with two objects in different spatial relations, and the remaining three splits are similar to those of LayoutBench.

We compare four models, covering both models with segmentation mask inputs (ControlNet [49]; we create segmentation masks by drawing bounding boxes with class-specific colors), and bounding box inputs (GLIGEN [23], ReCo [46], and our IterInpaint trained on COCO). For evaluation metric, we use layout accuracy in AP with a state-of-the-art object detector, YOLOv7 [43].

In Tab. 5, we show images generated by IterInpaint on four splits of LayoutBench-COCO (see appendix for more generation examples from other methods). Tab. 6 shows the layout accuracy of four evaluated models. The bounding box-based models outperform the segmentation mask-based model (ControlNet). Our IterInpaint achieves higher layout accuracy than baselines in all four splits, especially with a large margin in the combination split. The experiment results indicate the effectiveness of IterInpaint handling the challenging layouts.

6. Conclusion

We introduce LayoutBench, a diagnostic benchmark that systemically evaluates four spatial control skills of layout-guided image generation models: number, position, size, and shape. We show that recent layout-guided image generation methods do not generalize well on OOD layouts (*e.g*., many/large objects). Next, we propose IterInpaint, a new baseline that generates foreground and background regions step-by-step. In our detailed analysis of spatial control skills, IterInpaint has stronger generalizability than baselines on OOD layouts. We hope that our work facilitates future work on controllable image generation.

References

[1] Omri Avrahami, Thomas Hayes, Oran Gafni, Sonal Gupta, Yaniv Taigman, Devi Parikh, Dani Lischinski, Ohad Fried, and Xi Yin. SpaText: Spatio-Textual Representation for Controllable Image Generation. In *CVPR*, 2023. 2, 3

[2] Arpit Bansal, Hong-Min Chu, Avi Schwarzschild, Soumyadip Sengupta, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Universal guidance for diffusion models. In *CVPR Workshop*, 2023. 3

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-End Object Detection with Transformers. In *ECCV*, 2020. 4, 6

[4] Huiwen Chang, Han Zhang, Jarred Barber, AJ Maschinot, Jose Lezama, Lu Jiang, Ming-Hsuan Yang, Kevin Murphy, William T. Freeman, Michael Rubinstein, Yuanzhen Li, and Dilip Krishnan. Muse: Text-To-Image Generation via Masked Generative Transformers. In *ICML*, pages 1–22, 2023. 2

[5] 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

[6] Jaemin Cho, Abhay Zala, and Mohit Bansal. DALL-Eval: Probing the Reasoning Skills and Social Biases of Text-to-Image Generative Models. In *ICCV*, 2023. 4

[7] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. 3

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009. 3

[9] Wan-Cyuan Fan, Yen-Chun Chen, DongDong Chen, Yu Cheng, Lu Yuan, and Yu-Chiang Frank Wang. Frido: Feature pyramid diffusion for complex scene image synthesis. *arXiv preprint arXiv:2208.13753*, 2022. 1

[10] Stanislav Frolov, Tobias Hinz, Federico Raue, Jörn Hees, and Andreas Dengel. Adversarial Text-to-Image Synthesis: A Review. *Neural Networks*, 144:187–209, 2021. 3, 4, 7

[11] Stanislav Frolov, Avneesh Sharma, Jörn Hees, Tushar Karayil, Federico Raue, and Andreas Dengel. Attrlostgan: attribute controlled image synthesis from reconfigurable layout and style. In *DAGM German Conference on Pattern Recognition*, pages 361–375. Springer, 2021. 1

[12] Oran Gafni, Adam Polyak, Oron Ashual, Shelly Sheynin, Devi Parikh, and Yaniv Taigman. Make-A-Scene: Scene-Based Text-to-Image Generation with Human Priors. In *ECCV*, 2022. 2, 3

[13] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. In *NIPS*, 2014. 2

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016. 4

[15] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *NIPS*, 2017. 3, 6

[16] Tobias Hinz, Stefan Heinrich, and Stefan Wermter. Semantic Object Accuracy for Generative Text-to-Image Synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2020. 4

[17] Jonathan Ho and Tim Salimans. Classifier-Free Diffusion Guidance. In *NeurIPS Workshop*, 2021. 6

[18] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. In *NeurIPS*, 2020. 2

[19] Justin Johnson, Li Fei-Fei, Bharath Hariharan, C. Lawrence Zitnick, Laurens Van Der Maaten, and Ross Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*, 2017. 3

[20] 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. 2, 3, 6

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

[22] Yandong Li, Yu Cheng, Zhe Gan, Licheng Yu, Liqiang Wang, and Jingjing Liu. Bachgan: High-resolution image synthesis from salient object layout. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8365–8374, 2020. 1

[23] Yuheng Li, Haotian Liu, Qingyang Wu, Fangzhou Mu, Jianwei Yang, Jianfeng Gao, Chunyuan Li, and Yong Jae Lee. Gligen: Open-set grounded text-to-image generation. In *CVPR*, 2023. 2, 3, 8

[24] Zejian Li, Jingyu Wu, Immanuel Koh, Yongchuan Tang, and Lingyun Sun. Image synthesis from layout with locality-aware mask adaption. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13819–13828, 2021. 1

[25] 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. 4, 8

[26] Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo Numerical Methods for Diffusion Models on Manifolds. In *ICLR*, 2022. 6

[27] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *ICLR*, 2019. 6

[28] Elman Mansimov, Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Generating Images from Captions with Attention. In *ICLR*, 2016. 2

[29] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models. In *ICML*, 2022. 2

[30] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen

Krueger, Ilya Sutskever, Jong Wook, Kim Chris, Hallacy Aditya, Ramesh Gabriel, Goh Sandhini, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. In *ICML*, 2021. 4

[31] 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

[32] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical Text-Conditional Image Generation with CLIP Latents. *ArXiv*, 2204.06125, 2022. 2

[33] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *ICML*, 2016. 2

[34] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *NIPS*, 2015. 4

[35] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, pages 10684–10695, 2022. 1, 2, 3, 4, 5

[36] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding. In *NeurIPS*, 2022. 2

[37] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved Techniques for Training GANs. In *NIPS*, 2016. 3

[38] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, 2015. 2

[39] Wei Sun and Tianfu Wu. Image Synthesis From Reconfigurable Layout and Style. In *ICCV*, 2019. 3

[40] Wei Sun and Tianfu Wu. Image synthesis from reconfigurable layout and style. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10531–10540, 2019. 1

[41] Tristan Sylvain, Pengchuan Zhang, Yoshua Bengio, R. Devon Hjelm, and Shikhar Sharma. Object-Centric Image Generation from Layouts. In *AAAI*, pages 2647–2655, 2021. 3, 6

[42] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. In *CVPR*, 2016. 4

[43] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 8

[44] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks. In *CVPR*, 2018. 2

[45] Zuopeng Yang, Daqing Liu, Chaoyue Wang, Jie Yang, and Dacheng Tao. Modeling image composition for complex scene generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7764–7773, 2022. 1

[46] Zhengyuan Yang, Jianfeng Wang, Zhe Gan, Linjie Li, Kevin Lin, Chenfei Wu, Nan Duan, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. Reco: Region-controlled text-to-image generation. In *CVPR*, 2023. 1, 2, 3, 4, 5, 6, 8

[47] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, Ben Hutchinson, Wei Han, Zarana Parekh, Xin Li, Han Zhang, Jason Baldridge, and Yonghui Wu. Scaling Autoregressive Models for Content-Rich Text-to-Image Generation. *Transactions on Machine Learning Research*, 2022. 2

[48] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris Metaxas. Stack-GAN : Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks. In *ICCV*, 2017. 2

[49] Lvmin Zhang and Maneesh Agrawala. Adding Conditional Control to Text-to-Image Diffusion Models. In *ICCV*, 2023. 8

[50] Lisai Zhang, Qingcai Chen, Baotian Hu, and Shuoran Jiang. Text-guided neural image inpainting. In *ACM MM*, page 1302–1310, 2020. 5

[51] Bo Zhao, Lili Meng, Weidong Yin, and Leonid Sigal. Image Generation from Layout. In *CVPR*, 2019. 3

[52] Bo Zhao, Lili Meng, Weidong Yin, and Leonid Sigal. Image generation from layout. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8584–8593, 2019. 1

[53] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: Deformable Transformers for End-to-End Object Detection. In *ICLR*, 2021. 4