SLayR: Scene Layout Generation with Rectified Flow

Anonymous Author(s)
Affiliation
Address
email

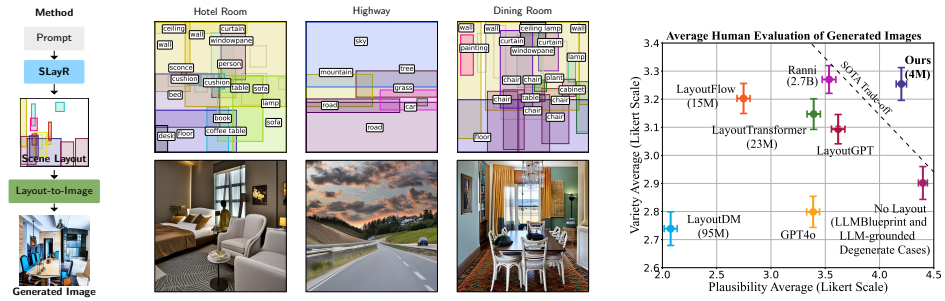


Figure 1: **Left:** We introduce **SLayR**, a method for scene layout generation via rectified flow. **Middle:** SLayR generates scene layouts for unconstrained prompts, which can be rendered using a layout-to-image generator. **Right:** Our method sets a new state of the art in generating more varied and yet plausible scenes than baselines, including LLMs.

Abstract

We introduce SLayR, **S**cene **Lay**out Generation with **R**ectified flow, a novel transformer-based model for text-to-layout generation, which can integrate into a complete text-to-image pipeline. SLayR addresses a domain in which current text-to-image pipelines struggle: generating scene layouts that are of significant variety and plausibility, when the given prompt is ambiguous and does not provide constraints on the scene. In this setting, SLayR surpasses existing baselines, including LLMs. To accurately evaluate the layout generation, we introduce a new benchmark suite, including numerical metrics and a carefully designed repeatable human-evaluation procedure that assesses the plausibility and variety of images that are generated. We show that our method sets a new state of the art for achieving high plausibility and variety simultaneously, while being at least 3× times smaller in the number of parameters.

1 Introduction

Recent advances in text-to-image modeling have focused on training denoising diffusion models [49, 14, 50] to generate images from a prompt encoding and image noise [42, 43, 44, 6, 61, 45], as well as incorporating finer-grained control modalities [15, 21, 37, 63, 33, 48, 34, 55]. Building upon these advancements, prior works have demonstrated the editability and interpretability advantages of a multistage text-to-layout-to-image model, where the user can view and manipulate an intermediate layout consisting of bounding boxes for object-level scene elements [26, 7, 66, 10, 60, 67, 1]. These works use LLMs as text-to-layout generators, and focus on parsing multi-object prompts (e.g. "two dogs next to a cat"). However, a closer inspection reveals that these models do not generate high variety (see fig. 1, right) or

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

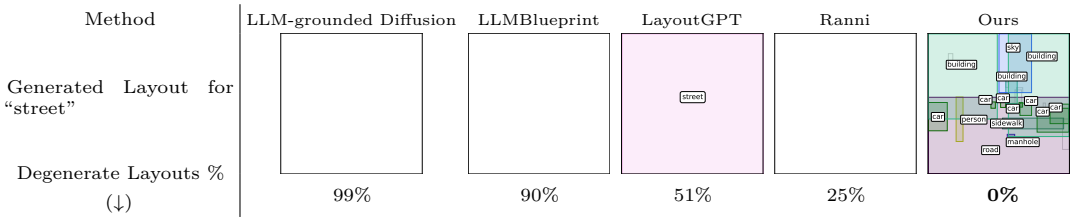


Figure 2: Degenerate layouts (where zero or one trivial bounding box is present) for the prompt "street" from LLM-grounded Diffusion [26], LLM Blueprint [9], LayoutGPT [7], and Ranni [8] vs. our layouts. The bottom shows percentages of degenerate layouts from our unconstrained prompt benchmark (See section 4). As visible, LLM approaches for constrained prompts do not generalize to the unconstrained setting.

collapse entirely (see Figure 2), when presented with prompts that have few constraints and leave a high degree of ambiguity. We see this as a critical problem: the models in these cases fail to present knowledge about the structure of scenes as they cannot rely on the prompt for specific information.

This motivates us to propose SLayR, a novel lightweight text-to-layout generation model for expanding unconstrained prompts (e.g. "a park", "a beach") into a variety of plausible scene layouts (see Figure 1, left and middle). Given a CLIP [41] embedding of a global scene prompt, we generate the layout using rectified flow [30], with a Diffusion Transformer (DiT) [38]. As unconstrained text-to-layout generation for general images has not been explored before, we assess our layout's plausibility and variety against both LLM-centric baselines and adapted UI/document generation. The experiments show that our method produces a very high variety, while achieving state-of-the-art plausibility in spatial arrangement.

Next, we combine our generated layouts with available layout-to-image generation models [52, 25, 56, 26] to create a complete text-to-image pipeline. We show that the generated images achieve the highest scores in CMMD [17], FID [13], KID [3], and HPSv2 [54] compared to the baselines. As true assessment of the image content is only possible by humans, we introduce a comprehensive and repeatable human-evaluation study. The ratings show that our model yields the state-of-the-art trade-off in generating images that are both diverse and plausible. In addition, our pipeline is significantly more lightweight than baselines and can be conditioned on partial layouts and directional constraints, while also providing the ability to edit layouts.

In summary, our contributions are: **1)** we introduce the first model for rectified flow-based text-to-layout generation and show that it produces a large variety of highly plausible layouts for challenging unconstrained prompts, **2)** we establish a well-designed human-evaluation study that can be repeated by others, and **3)** demonstrate that integrating our method into a complete text-to-layout-to-image pipeline yields state-of-the-art in achieving variety and plausibility together. See our supplement to access source code.

2 Related Work

LLMs in Scene Layout Generation. Prior works in 2D layout generation leverage LLMs to parse multi-object prompts into layouts, typically leveraging in-context learning [26, 9, 7, 8]. Querying these models with unconstrained prompts frequently yields degenerate solutions without meaningful layout information (See fig. 2). Given that LLM-grounded Diffusion [26] and LLM Blueprint [9] degenerate in 90% or more cases, we do not evaluate them further. Results on LayoutGPT [7] and Ranni [8] are provided. To control for the shift to the unconstrained prompt domain, we also adapt the prompt template from [26] with in-context examples from our domain, and encouragement of chain-of-thought reasoning [53], to meaningfully assess an LLM's capabilities for this task. For the underlying LLM, we use GPT4o [35]

Adapting UI Generation. Our task of scene layout generation is distinct from User Interface (UI) generation: scene and object captions are from open sets, whereas UI layouts lack global captions and have labels from a small fixed set. Nevertheless, they can serve as interesting baselines, and we adapt several of these models using their conditional generation

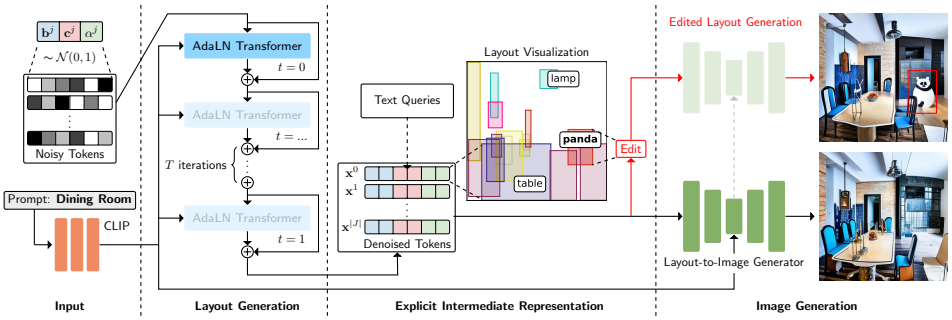


Figure 3: **Method Overview.** Our layout generation model takes a set of noisy tokens and a prompt encoded as a global CLIP embedding as input. The tokens are partitioned into bounding box information $\mathbf{b}^j$, reduced CLIP embeddings $\mathbf{c}^j$, and opacities α^j, with j being the object index. The tokens are then subsequently denoised from $t = 0$ to $t = 1$ using a transformer. For visualization purposes, the user can query the generated layout with labels and edit boxes by adding, moving or removing them. Finally, the generated layout is passed through an off-the-shelf layout-to-image generator.

capabilities. We use LayoutTransformer [12] as a representative for autoregressive transformer approaches, which completes a partial sequence of object bounding boxes to form an image layout. LayoutFormer++ [18] extends LayoutTransformer with added conditioning, but this is not the focus of our assessment of adapted UI generation, and thus it is a redundant baseline. We also adapt LayoutDM [16] and LayoutFlow [11] as representative baselines for diffusion-based methods for UI generation [62, 4, 23]. For GAN-based approaches [24], while LayoutGAN++ [20] supports inter-bounding-box relationships, the Lagrange multiplier constraint formulation cannot be adapted to support global conditioning. In contrast to our method, UI generation models by design do not extend into the open world scenario.

Rectified Flow. Diffusion modeling has inspired numerous variants and improvements, one of which is rectified flow [30]. Prior works on the related text-to-image generation task [31, 6]. An initial ablation on DDIM [50], indicates that rectified flow outperforms traditional diffusion approaches [14] in this setting. See the supplement for details.

Layout-to-Image Generation. We demonstrate that SLayR integrates well into downstream conditional diffusion models to form a complete text-to-image pipeline, with the added benefits of an interpretable and controllable intermediate layout phase. To control for the effect which the image generator has on the final generated image, we evaluate our layouts across multiple layout-to-image models. Although there are a wide variety of such models, [5, 59, 64, 51, 2, 57] we select four which are publically available and have been used successfully with LLM-driven layouts [26, 7] or have shown SOTA performance: InstanceDiffusion [52], GLIGEN [25], BoxDiff [56], and LMD+ [26].

3 Method

The central part of our work is the text-to-layout generation module, which we combine with the existing layout-to-image generators to form a complete text-to-image pipeline. An overview is provided in fig. 3, and we explain the details below.

Layout Representation. We start with defining a scene representation as the basis for our generative architecture. A training sample $(\mathbf{x}, P)$ is composed of a global image caption prompt P and a set of J object tokens $\mathbf{x} = \{\mathbf{x}^j \in \mathbb{R}^{d+5}\}_{j \in J}$. The token representation of any single object is composed of

$$\mathbf{x}^j = (\mathbf{b}^j \parallel \mathbf{c}^j \parallel \alpha^j), \tag{1}$$

where $\mathbf{b}^j = (x^j, y^j, w^j, h^j) \in \mathbb{R}^4$ encodes the bounding box coordinates, $\mathbf{c}^j \in \mathbb{R}^d$ is a PCA-reduced CLIP [41] embedding, and $\alpha^j \in \mathbb{R}$ is an opacity value that defines the existence of a specific bounding box.

Rectified Flow Preliminaries. We briefly recap the basics of rectified flow introduced in [30]. Let I be a set of training sample indices and $\{\mathbf{x}_i\}_{i \in I}$ the ground-truth samples

whose distribution we would like to learn using our model v. We linearly interpolate between Gaussian noise $\mathbf{x}_i(0)$ and samples $\mathbf{x}_i(1) \equiv \mathbf{x}_i$ across timesteps $t \in [0, 1]$ as follows:

$$\mathbf{x}_i(t) = (1-t) \cdot \mathbf{x}_i(0) + t \cdot \mathbf{x}_i(1)\,. \tag{2}$$

The model v is trained to take $(\mathbf{x}_i(t), t)$ as input and to predict the derivative of the path between $\mathbf{x}_i(0)$ and $\mathbf{x}_i(1)$, which according to Equation 2 is $\mathbf{x}_i(1) - \mathbf{x}_i(0)$. The training objective is:

$$\min_v \int_0^1 \mathbb{E}_i[||(\mathbf{x}_i(1) - \mathbf{x}_i(0)) - v(\mathbf{x}_i(t), t)||^2]dt \tag{3}$$

and is optimized with stochastic gradient descent. This optimization is carried out across all available samples of the ground-truth distribution. Following [30], noisy values $\mathbf{x}_i(0)$ are resampled at each epoch. The end result is a network v, which is effective at predicting the direction from a noisy sample at an intermediate timestep towards the target distribution. Since a single evaluation may be noisy, the inference is performed by integrating over T timesteps:

$$\mathbf{x}_i(1) = \mathbf{x}_i(0) + \sum_{t=1}^{T} v(\mathbf{x}_i(\frac{t-1}{T}), \frac{t}{T}) \cdot \frac{1}{T}\,. \tag{4}$$

Our Model Architecture. Our rectified flow model is built from multihead AdaLN transformer blocks, which can process tokens $\{\mathbf{x}_i^j\}_{j \in J}$ to iteratively denoise them [38]. The timestep t, bounding box coordinates $\mathbf{b}_i^j(t)$, and opacity values $\alpha_i^j(t)$ are sinusoidally encoded. The timestep t and a linear projection of the global P_i's CLIP encoding are passed as conditioning of the adaptive layer normalization of the transformer blocks. The tokens represent the objects in the layout and are processed all at once.

Inference begins at $t = 0$ with the set of tokens $\{\mathbf{x}_i^j(t)\}_{j \in J} \equiv \{\mathbf{x}_i^j(0)\}_{j \in J}$ initialized from Gaussian noise. Our model then iteratively processes and updates the tokens from $t = 0$ to $t = 1$ over T iterations using eq. (4) based on the global prompt conditioning P_i. We project this output back to the dimension of $\mathbf{x}_i^j(t)$ before sinusoidal encoding, in order for the module to serve as the rate of change of $\mathbf{x}_i^j(t)$. A single inference step can be summarized as:

$$\{\mathbf{x}_i^j(t)\}_{j \in J} \leftarrow \{\mathbf{x}_i^j(t - \frac{1}{T})\}_{j \in J} + v(\{\mathbf{x}_i^j(t - \frac{1}{T})\}_{j \in J}, t - \frac{1}{T}, P_i) \cdot \frac{1}{T}\,, \tag{5}$$

Following eq. (5) until $t = 1$ yields the final layout $\{\mathbf{x}_i^j(1)\}_{j \in J}$ that contains PCA-reduced CLIP embeddings, bounding boxes, and opacities. Tokens with $\alpha_i^j(1) < 0.5$ are considered unused and discarded, please see the supplement for further explanation. For image generation, we unproject each $\mathbf{c}_i^j(1)$ from the PCA space back into the CLIP feature space and pass the embeddings directly into the downstream image generation module.

For visualization of the layouts, we follow the common practice when interpreting visual representations in natural language [19, 39] and decode CLIP embeddings to text by comparing them to label queries from the user, and selecting the closest query in the embedding space. In the supplement, we explain the RePaint [32, 46] technique for rectified flow to enable *partial layout conditioning*. This enables our model to be guided by partial layouts where only some boxes or labels are given (see fig. 6). We additionally show how we can impose inter-bounding box positional constraints (i.e, place A to the *left* of B) by adding a directional drift on the bounding boxes during inference. The ability to control our model through these conditions allows it to also work in concert with an LLM to handle complex prompts, where the role of the LLM is to extract the constraints from the prompt, and our method takes care of generating the remaining unspecified scene details.

Training. To construct a training sample from the ground-truth image layout i, we create $\mathbf{c}_i^j$ and $\mathbf{b}_i^j$ for each bounding box j, and initialize α_i^j to 1. To ensure a consistent amount of tokens, we pad the samples by adding tokens with $\alpha_i^j = 0$ and $\mathbf{b}_i^j = 0$, and $\mathbf{c}_i^j$ to the null string embedding. We now treat $\{\mathbf{x}_i^j\}_{j \in J} \equiv \{\mathbf{x}_i^j(1)\}_{j \in J}$, sample $\{\mathbf{x}_i^j(0)\}_{j \in J}$ from Gaussian noise, draw t uniformly from $[0, 1]$, and compute the set of tokens $\{\mathbf{x}_i^j(t)\}_{j \in J}$ by adapting the

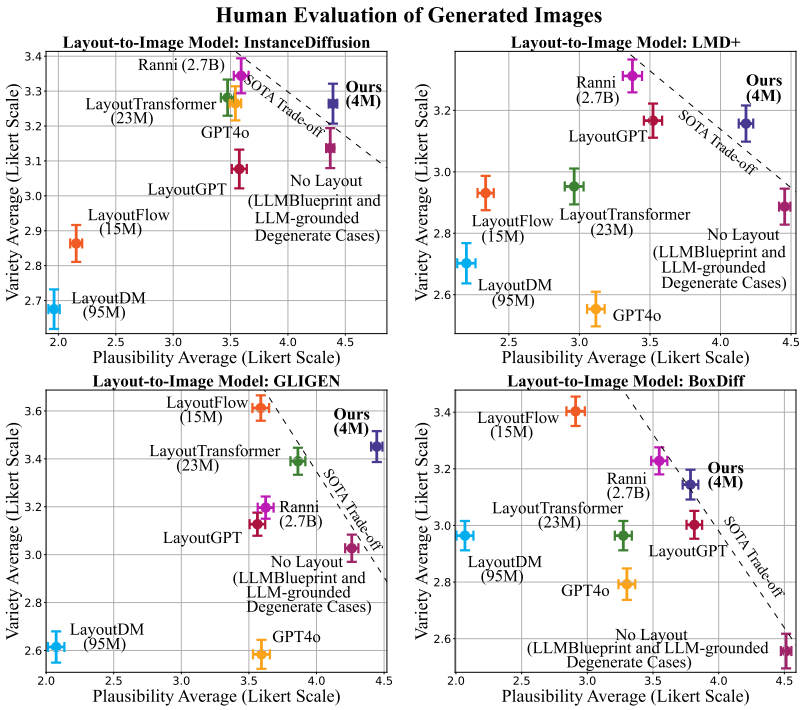


Figure 4: **Human Survey Results.** Our method offers an equal or superior trade-off between plausibility and variety across all measured layout-to-image generators, while being a much smaller model. The error bars indicate standard error.

formula from eq. (2), which are then passed to the model as input. We refer to the output of the model as $v(\{\mathbf{x}_i^j(t)\}_{j\in J}, t, P_i)$ and compute the training loss derived from eq. (3):

$$\mathcal{L} = \sum_{i\in I, j\in J} ||\mathbf{x}_i^j(1) - \mathbf{x}_i^j(0) - v(\{\mathbf{x}_i^j(t)\}_{j\in J}, t, P_i)_j||^2 . \tag{6}$$

Human Evaluation. Given the novelty of our problem domain, we argue that human evaluation is most reliable for assessing the plausibility and variety of layouts and therefore introduce a human-evaluation study which can be repeated by others. Assessing human opinions for these criteria directly on layouts is challenging: the evaluators require time to understand the layout diagrams and explain them, and furthermore, assessments are hard to make without actually seeing the image. Following the design principles presented by Otani *et al.* [36] in their work on human evaluation of text-to-image generation: 1) *the (evaluation) task should be simple*, and 2) *results should be interpretable*. Therefore, we show participants only images, and omit the underlying image layouts entirely, which may take effort to understand. To make the results interpretable, participants rate these images for their *plausibility* and *variety* on a Likert scale (as recommended in Otani *et al.* [36]) from 1 to 5. Image qualities that are assessed in other studies (for example, the overall quality and aesthetic appeal of the image in Liang *et al.* [27]) are highly dependent on the conditioned image generator. Therefore, we consider these misleading for our case.

The study is approved by the Ethics Review Board of our institution and complies with local wage regulations. To keep the cost of a survey below 250 USD, we survey 60 participants, who each assess four text-to-layout generation methods at once, each providing ten plausibility questions and ten variety ratings. To increase the stability of the results and test on a larger sample set, each rating is for a collection of three images from the same prompt. The subset of collections, as well as the order they are displayed to the participant, are randomized to control for any potential effects of a fixed ordering. An expanded explanation of our survey design, including the text instructions and screenshots of the survey, can be found in the supplemental material.

Figure 5: **Qualitative comparison** (Best viewed up close). Layout objects that are depicted in the generated image are highlighted and labeled. From a visual inspection, having no layout produces scenes of little variation in content. LayoutFlow's layouts do not appear to capture scene structure. GPT4o's layouts lack variety. Layout Transformer produces layouts with implausible arrangements of objects, leading to images which do not depict the global prompt accurately. Our method creates plausible and varied layouts, leading to images that are diverse and look realistic. These observations are supported by our human evaluation in fig. 4. Zoomed-in versions of these layouts for printing are available in the supplemental.

4 Experiments

Dataset. We test our method's ability to learn a variety of plausible scene layouts by both training and evaluating on the full ADE20K dataset [65], which contains approximately 27K images and ground-truth layouts for indoor and outdoor scenes, and a rich collection of object arrangements. The sample captions reflect the scene category with no additional constraints (e.g., "beach", "lecture room"). We use the top 30 largest bounding boxes per sample, as this is the default maximum number of bounding boxes supported by InstanceDiffusion [52] and we pad samples with fewer bounding boxes. For evaluation, we use the 15 highest

represented categories and add in five randomly selected categories to include the dataset's long tail distribution. For each evaluated model, we generate 30 layouts for all 20 selected prompts, and an image conditioned on each layout and corresponding global prompt. The size of this collection of images makes it feasible to assess the results with human evaluation.

Model	CMMD (↓)	FID (↓)	KID (10^{-2}) (↓)	HSPv2 (↑)	Image Reward (↑)	VQA (↑)
LayoutFlow	0.25	0.80	0.88	<u>0.23</u>	−1.01	0.80
LayoutDiffusion	0.40	1.08	1.99	0.19	−2.11	0.34
LayoutTransformer	<u>0.06</u>	<u>0.44</u>	<u>0.30</u>	<u>0.23</u>	−1.00	0.75
GPT4o	0.09	0.94	0.45	**0.25**	−0.51	0.88
Ranni	0.07	0.71	<u>0.30</u>	**0.25**	**−0.34**	<u>0.90</u>
LayoutGPT	0.29	2.83	1.76	**0.25**	−0.26	**0.93**
Ours	**0.03**	**0.17**	**0.16**	**0.25**	<u>−0.32</u>	0.88

Table 1: **Image Metrics Comparison.** We evaluate traditional metrics and compare the images generated from layouts of different layout generators. To avoid biases of the image generator, we show the best score among the layout-to-image generators InstanceDiffusion [52], GLIGEN [25], BoxDiff [56], and LMD+ [26] for each layout generator. Our method achieves strong or state-of-the-art numbers for measured metrics. Although their metrics are strong, Ranni and LayoutGPT are susceptible to degenerate solutions (see fig. 2)

Baselines. We compare our method against prior works which are capable of unconstrained layout generation. For LLM-baselines, we evaluate against LayoutGPT [7] and Ranni [8], but discard LLM-grounded Diffusion [26] and LLM Blueprint [9], as these give degenerate cases in 90%+ of measured cases in our domain (see fig. 2). To see if LLM performance can be improved with proper in-context examples, we adapt the template from [26] with relevant in-context-learning examples from ADE20K. For the underlying LLM, we select the large-scale LLM GPT4o [35], and refer to this baseline simply as GPT4o. The full template is in the supplement. We test aginst the UI generators LayoutTransformer [12], LayoutDM [16] and LayoutFlow [11] by treating the global caption as a scene-wide bounding box and conditioning the model on this bounding box during inference. When training models, we stuck to their respective provided training settings.

Human Evaluation. As shown in fig. 4, our model achieves a state-of-the-art balance in image plausibility and variety across all measured layout-to-image generators: InstanceDiffusion [52], GLIGEN [25], BoxDiff [56], and LMD+ [26]. The error bars indicate standard error ($s = \frac{\sigma}{\sqrt{n}}$) of the mean human rating, calculated using `numpy` We assume normally distributed errors. display the approximate number of model parameters added to the full text-to-layout-to-image pipeline by the layout generators that can be locally run. Our model is the smallest by over a factor 3.

Visual Results. We provide a qualitative overview of the generated layouts and the final images in fig. 5, with InstanceDiffusion [52] as the layout-to-image model. We label bounding boxes by querying with all text labels present within ADE20K. From a visual inspection, LayoutTransformer struggles with arranging objects in spatially plausible manner. GPT4o layouts appear somewhat flat, while struggling to make a variety of layouts. Our method appears to produce both plausible and diverse images across a range of global prompts of indoor and outdoor settings.

Generated Image Metrics. We compute established image generation metrics CMMD [17], FID [13], KID [3], VQA [29], HPSv2 [54], and ImageReward [58]. CMMD, FID and KID compare the distribution of generated images with a ground-truth distribution, while VQA, HSPv2 and ImageReward assess general image quality and alignment with a global caption. Since the conditioned image generator may itself lead to biases in image generation quality, for CMMD, FID, and KID, we establish the ground-truth images by running the layout-to-image generator on the ground-truth layouts. For each layout generator, we display the optimal score over the possible combinations of layout and image generator ([52, 25, 56, 26]). Images from degenerate layouts from Ranni and LayoutGPT are discarded to more clearly assess the layout's influence. The results are shown in table 1, with state-of-the-art performance in CMMD, FID, KID and HSPv2, and strong results in ImageReward and VQA.

Scene Layout Metrics, and Speed. We consider how to best assess scene layouts for unconstrained prompts. The traditional UI generation metrics of Alignment [22] and

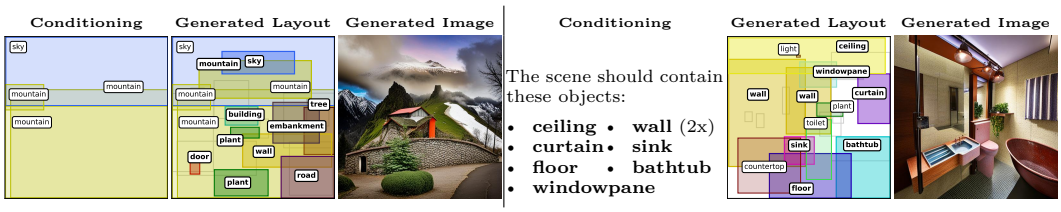


Figure 6: **Disentangled Generation.** Disentangled generation for scenes with the prompt *Snowy Mountain* with a partial layout (**Left**), and *Bathroom* with a *bag of words* (**Right**).

Overlap [24] scores are not salient, as real world images often have misaligned or overlapping bounding boxes. Likewise, the layout-FID [13] metric requires a layout-GAN discriminator to compute, which we do not have in this new domain. We compute a standard mIoU [20] averaged across sampled scene categories. To provide a more complete evaluation, we introduce metrics aimed to quantify a generated layout's *plausibilty* and *variety* that we describe in full in the supplementary material. We measure the model's generation time on batches of 30 layout samples on an Nvidia A6000 GPU with 32 AMD Ryzen 9 5950X CPUs, 125 GB RAM, except for GPT4o that is accessed through an API. Numerical results are provided in the supplement. Notably, we achieve the highest performance in positional likelihood (how plausibly objects are arranged) and mIoU. Our method ranks second in speed only to LayoutFlow, but we observe no definitive improvement in its layout statistics when the number of inference steps are raised to match our model's speed.

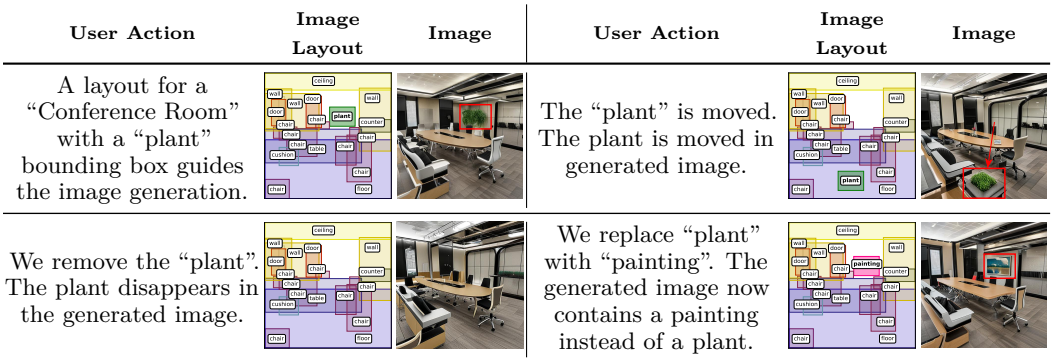


Figure 7: **Editing.** We show how our pipeline enables user editing of images by altering the intermediate scene layout representation. Individual objects can be easily moved, removed, and replaced.

Additional Model Features. We breifly highlight qualities of SLayR which make it appealing to use: In fig. 6, we show examples of our model's performance in different partial layout generation settings. This feature gives users even more fine-grained control over the image generation process. Additionally, we demonstrate how a text-to-layout-to-image pipeline allows for editing of generated images in fig. 7. This is accomplished through modifying the intermediate scene layout, and rerunning layout-to-image generator with the original seed and global prompt.

5 Conclusion

We have introduced a text-to-layout model, incorporating it into a text-to-image pipeline with an intermediate and controllable layout representation. With a substantially smaller model, we can generate images with a start-of-the-art balance in plausibility and variety, while achieving high or state-of-the-art performance in generated image quality metrics among competing baselines. In addition, we have introduced a suite of metrics for the new task of scene layout generation, with which we established the foundation to explore image generation pipelines with explicit intermediate layouts in the future.

References

[1] Rio Aguina-Kang, Maxim Gumin, Do Heon Han, Stewart Morris, Seung Jean Yoo, Aditya Ganeshan, R. Kenny Jones, Qiuhong Anna Wei, Kailiang Fu, and Daniel Ritchie. Open-universe indoor scene generation using llm program synthesis and uncurated object databases, 2024.

[2] Omer Bar-Tal, Lior Yariv, Yaron Lipman, and Tali Dekel. Multidiffusion: Fusing diffusion paths for controlled image generation, 2023.

[3] Mikołaj Bińkowski, Danica J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans, 2021.

[4] Shang Chai, Liansheng Zhuang, and Fengying Yan. Layoutdm: Transformer-based diffusion model for layout generation, 2023.

[5] Minghao Chen, Iro Laina, and Andrea Vedaldi. Training-free layout control with cross-attention guidance, 2023.

[6] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis, 2024.

[7] Weixi Feng, Wanrong Zhu, Tsu jui Fu, Varun Jampani, Arjun Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. Layoutgpt: Compositional visual planning and generation with large language models, 2023.

[8] Yutong Feng, Biao Gong, Di Chen, Yujun Shen, Yu Liu, and Jingren Zhou. Ranni: Taming text-to-image diffusion for accurate instruction following, 2024.

[9] Hanan Gani, Shariq Farooq Bhat, Muzammal Naseer, Salman Khan, and Peter Wonka. Llm blueprint: Enabling text-to-image generation with complex and detailed prompts, 2024.

[10] Gege Gao, Weiyang Liu, Anpei Chen, Andreas Geiger, and Bernhard Schölkopf. Graphdreamer: Compositional 3d scene synthesis from scene graphs, 2024.

[11] Julian Jorge Andrade Guerreiro, Naoto Inoue, Kento Masui, Mayu Otani, and Hideki Nakayama. Layoutflow: Flow matching for layout generation, 2024.

[12] Kamal Gupta, Justin Lazarow, Alessandro Achille, Larry Davis, Vijay Mahadevan, and Abhinav Shrivastava. Layouttransformer: Layout generation and completion with self-attention, 2021.

[13] 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, 2018.

[14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

[15] Drew A. Hudson, Daniel Zoran, Mateusz Malinowski, Andrew K. Lampinen, Andrew Jaegle, James L. McClelland, Loic Matthey, Felix Hill, and Alexander Lerchner. Soda: Bottleneck diffusion models for representation learning, 2023.

[16] Naoto Inoue, Kotaro Kikuchi, Edgar Simo-Serra, Mayu Otani, and Kota Yamaguchi. Layoutdm: Discrete diffusion model for controllable layout generation, 2023.

[17] Sadeep Jayasumana, Srikumar Ramalingam, Andreas Veit, Daniel Glasner, Ayan Chakrabarti, and Sanjiv Kumar. Rethinking fid: Towards a better evaluation metric for image generation, 2024.

[18] Zhaoyun Jiang, Jiaqi Guo, Shizhao Sun, Huayu Deng, Zhongkai Wu, Vuksan Mijovic, Zijiang James Yang, Jian-Guang Lou, and Dongmei Zhang. Layoutformer++: Conditional graphic layout generation via constraint serialization and decoding space restriction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18403–18412, 2023.

[19] Justin Kerr, Chung Min Kim, Ken Goldberg, Angjoo Kanazawa, and Matthew Tancik. Lerf: Language embedded radiance fields, 2023.

[20] Kotaro Kikuchi, Edgar Simo-Serra, Mayu Otani, and Kota Yamaguchi. Constrained graphic layout generation via latent optimization. In *Proceedings of the 29th ACM International Conference on Multimedia*, page 88–96. ACM, 2021.

[21] Mingi Kwon, Jaeseok Jeong, and Youngjung Uh. Diffusion models already have a semantic latent space, 2023.

[22] Hsin-Ying Lee, Lu Jiang, Irfan Essa, Phuong B Le, Haifeng Gong, Ming-Hsuan Yang, and Weilong Yang. Neural design network: Graphic layout generation with constraints, 2020.

[23] Elad Levi, Eli Brosh, Mykola Mykhailych, and Meir Perez. Dlt: Conditioned layout generation with joint discrete-continuous diffusion layout transformer, 2023.

[24] Jianan Li, Jimei Yang, Aaron Hertzmann, Jianming Zhang, and Tingfa Xu. Layoutgan: Generating graphic layouts with wireframe discriminators, 2019.

[25] 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, 2023.

[26] Long Lian, Boyi Li, Adam Yala, and Trevor Darrell. Llm-grounded diffusion: Enhancing prompt understanding of text-to-image diffusion models with large language models, 2024.

[27] Youwei Liang, Junfeng He, Gang Li, Peizhao Li, Arseniy Klimovskiy, Nicholas Carolan, Jiao Sun, Jordi Pont-Tuset, Sarah Young, Feng Yang, Junjie Ke, Krishnamurthy Dj Dvijotham, Katie Collins, Yiwen Luo, Yang Li, Kai J Kohlhoff, Deepak Ramachandran, and Vidhya Navalpakkam. Rich human feedback for text-to-image generation, 2024.

[28] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[29] Zhiqiu Lin, Deepak Pathak, Baiqi Li, Jiayao Li, Xide Xia, Graham Neubig, Pengchuan Zhang, and Deva Ramanan. Evaluating text-to-visual generation with image-to-text generation, 2024.

[30] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow, 2022.

[31] Xingchao Liu, Xiwen Zhang, Jianzhu Ma, Jian Peng, and Qiang Liu. Instaflow: One step is enough for high-quality diffusion-based text-to-image generation, 2024.

[32] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models, 2022.

[33] Grace Luo, Trevor Darrell, Oliver Wang, Dan B Goldman, and Aleksander Holynski. Readout guidance: Learning control from diffusion features, 2024.

[34] Rameshwar Mishra and A V Subramanyam. Image synthesis with graph conditioning: Clip-guided diffusion models for scene graphs, 2024.

[35] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, et al. Gpt-4 technical report, 2024.

[36] Mayu Otani, Riku Togashi, Yu Sawai, Ryosuke Ishigami, Yuta Nakashima, Esa Rahtu, Janne Heikkilä, and Shin'ichi Satoh. Toward verifiable and reproducible human evaluation for text-to-image generation, 2023.

[37] Yong-Hyun Park, Mingi Kwon, Jaewoong Choi, Junghyo Jo, and Youngjung Uh. Understanding the latent space of diffusion models through the lens of riemannian geometry, 2023.

[38] William Peebles and Saining Xie. Scalable diffusion models with transformers, 2023.

[39] Minghan Qin, Wanhua Li, Jiawei Zhou, Haoqian Wang, and Hanspeter Pfister. Langsplat: 3d language gaussian splatting, 2024.

[40] Qualtrics. Qualtrics xm platform, 2024. Computer software.

[41] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

[42] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation, 2021.

[43] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022.

[44] 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, 2022.

[45] Axel Sauer, Frederic Boesel, Tim Dockhorn, Andreas Blattmann, Patrick Esser, and Robin Rombach. Fast high-resolution image synthesis with latent adversarial diffusion distillation, 2024.

[46] Philipp Schröppel, Christopher Wewer, Jan Eric Lenssen, Eddy Ilg, and Thomas Brox. Neural point cloud diffusion for disentangled 3d shape and appearance generation, 2024.

[47] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. Laion-5b: An open large-scale dataset for training next generation image-text models, 2022.

[48] Guibao Shen, Luozhou Wang, Jiantao Lin, Wenhang Ge, Chaozhe Zhang, Xin Tao, Yuan Zhang, Pengfei Wan, Zhongyuan Wang, Guangyong Chen, Yijun Li, and Ying-Cong Chen. Sg-adapter: Enhancing text-to-image generation with scene graph guidance, 2024.

[49] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015.

[50] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022.

[51] Tristan Sylvain, Pengchuan Zhang, Y. Bengio, R Devon Hjelm, and Shikhar Sharma. Object-centric image generation from layouts, 2020.

[52] Xudong Wang, Trevor Darrell, Sai Saketh Rambhatla, Rohit Girdhar, and Ishan Misra. Instancediffusion: Instance-level control for image generation, 2024.

[53] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.

[54] Xiaoshi Wu, Yiming Hao, Keqiang Sun, Yixiong Chen, Feng Zhu, Rui Zhao, and Hongsheng Li. Human preference score v2: A solid benchmark for evaluating human preferences of text-to-image synthesis, 2023.

[55] Yang Wu, Pengxu Wei, and Liang Lin. Scene graph to image synthesis via knowledge consensus. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(3):2856–2865, 2023.

[56] Jinheng Xie, Yuexiang Li, Yawen Huang, Haozhe Liu, Wentian Zhang, Yefeng Zheng, and Mike Zheng Shou. Boxdiff: Text-to-image synthesis with training-free box-constrained diffusion, 2023.

[57] Zhexiao Xiong, Wei Xiong, Jing Shi, He Zhang, Yizhi Song, and Nathan Jacobs. Groundingbooth: Grounding text-to-image customization, 2024.

[58] Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation, 2023.

[59] 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, 2022.

[60] Xuening Yuan, Hongyu Yang, Yueming Zhao, and Di Huang. Dreamscape: 3d scene creation via gaussian splatting joint correlation modeling, 2024.

[61] Chenshuang Zhang, Chaoning Zhang, Mengchun Zhang, and In So Kweon. Text-to-image diffusion models in generative ai: A survey, 2023.

[62] Junyi Zhang, Jiaqi Guo, Shizhao Sun, Jian-Guang Lou, and Dongmei Zhang. Layoutdiffusion: Improving graphic layout generation by discrete diffusion probabilistic models, 2023.

[63] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models, 2023.

[64] Bo Zhao, Lili Meng, Weidong Yin, and Leonid Sigal. Image generation from layout, 2019.

[65] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset, 2018.

[66] Xiaoyu Zhou, Xingjian Ran, Yajiao Xiong, Jinlin He, Zhiwei Lin, Yongtao Wang, Deqing Sun, and Ming-Hsuan Yang. Gala3d: Towards text-to-3d complex scene generation via layout-guided generative gaussian splatting, 2024.

[67] Başak Melis Öcal, Maxim Tatarchenko, Sezer Karaoglu, and Theo Gevers. Sceneteller: Language-to-3d scene generation, 2024.

Supplementary Material

The supplementary material is structured as follows. First, we present the full details of the human evaluation study performed to judge the generation quality in appendix A. Next, we introduce details about our partial conditioning procedure in appendix B. We provide the formulae and reasoning for our novel layout metrics in appendix C. We further provide detailed information about training data and hyperparameters in appendix D, provide access to our source code in appendix E discuss limitations in appendix F, broader impacts in appendix G, safe guards in appendix H, licenses in appendix I, discussions about LayoutTransformer and GPT4o temperatures in appendix J and appendix K, respectively, and the GPT4o query template in appendix L. In appendix O, we compare editing generated images in a text-to-layout-to-image pipeline against a drag-editing method. In appendix P, we analyze the distribution of token opacities that our model produces to justify $\alpha_i^j(1) < 0.5$ as our token discarding threshold Last, we provide a comparison between rectified flow and DDIM in appendix M and high-resolution results in appendix N and appendix Q.

At the end, we supply the checklist (Jump to appendix Q)

A Human Evaluation Details

Study Goal. Although our method achieves optimal performance in table 1, we aim to confirm that these metrics, which were designed for measuring the quality of text-to-image models, are applicable to text-to-layout-to-image models. We also want to control for the effect which the layout-to-image model could have on the final quality, and assess how effective the underlying layouts are in the image generation process. To this end, we provide a human-evaluation study that can be repeated by others.

In general, we want a text-to-layout model to generate layouts that appear plausible and are also of a large variety. However, assessing human opinions for these criteria directly on layouts is challenging: the evaluators require time to understand the layout diagrams and explain them, and furthermore, assessments are hard to make without actually seeing the image. To make the study effective, we measure the effect of our model on the downstream generated images. Image qualities that are assessed in other studies (for example, the overall quality and aesthetic appeal of the image in Liang *et al.* [27]) are highly dependent on the conditioned image generator. Therefore, we consider these misleading for our case and introduce a suitable study in the following.

Design Principles. We follow the design principles presented by Otani *et al.* [36] in their work on human evaluation of text-to-image generation: 1) *the (evaluation) task should be simple*, and 2) *results should be interpretable*. Following these principles, we show participants only images, and omit the underlying image layouts entirely, which may take some effort to understand. To make the results interpretable, participants rate these images for their plausibility and variety on a Likert scale (as specifically recommended in Otani *et al.* [36]) from 1 to 5. This way, average ratings for different layout generation models can be meaningfully compared to each other, which would be more difficult in other systems (e.g. using non-numbered ranking). To ensure cost efficiency, our survey must be small enough that the data can be collected quickly and repeatedly throughout the model development, and thus we show participants collections of images rather than singletons. We kept all of these constraints in mind when designing our study, which is explained in further detail below.

Study Description. Our study was developed using Qualtrics [40], a standard survey platform. Each participant answers ten plausibility questions and ten variety questions, meaning they rate 80 image collections in total. Each collection contains three images. We survey 60 participants. The prompts, image collection index, and the order in which the collections are displayed to participants is randomized to control for any potential effects of a fixed ordering.

Survey data is selected as described in section 4. As shown in fig. 9, each survey question shows collections of three images from each of the four methods listed above, where every image on the screen has the same global prompt. Given the instructions from fig. 8, the

participant must rate each collection for either their plausibility or variety. Ratings are on a Likert scale (1 to 5, where 1 corresponds to very implausible/very low variance, and 5 corresponds to very plausible/very high variance). For plausibility, we instructed participants to consider the overall realism of the collection, as well as how effectively it depicts the global text prompt. For variety, we instructed users to consider the spatial arrangement of objects in an image and implied camera angle in addition to overall image appearance.

Participant Selection and Ethics. Participants were recruited through Connect CloudResearch, a crowdsourcing service built on Amazon Mechanical Turk that implements rigorous quality control procedures to enhance the reliability of the participant pool in line with the study recommendations given by Otani *et al.* [36]. The study was approved by the Ethics Review Board of our institution, ensuring compliance with ethical standards. Prior to engaging in the tasks, all participants were informed about the content of the survey, and then provided their informed consent. We did not see any risks that could be incurred by participation in the survey, and therefore had no risks to disclose. The study was designed to be completed within 15 minutes by each participant, who were compensated at an hourly rate of 13.02 USD, complying with local wage regulations. This results in a total cost of 245 USD per run to assess four text-to-layout generation methods at once. Participants were anonymized, and we did not collect any personally-identifiable information.

Section 1: Plausibility

> "For the following section of the survey, you will be asked to rate collections of images based on how **plausible** they appear to be, from **very implausible** to **very plausible**. An image is considered plausible if objects within the image are **realistically and organically** placed, and it is a **reasonable match to the presented caption**. The images do not have to be photorealistic to be considered plausible. You will perform ratings on 10 categories of images, and each page will contain 4 collections that you must rate separately."

Section 2: Variety

> "For the following section of the survey, you will be asked to rate collections of images based on their perceived **variance**, from **very low variance** to **very high variance**. When judging the variance, consider criteria such as the differences in the **spatial arrangement** of objects, the differences in **camera perspective**, and the differences in the **overall image appearance** across the collection. You will perform ratings on 10 categories of images, and each page will contain 4 collections that you must rate separately."

Figure 8: Full instructions to participants for both sections of the survey. Our instructions clearly define the task and give users detailed information on what to assess

B Implementation of Partial Conditioning

We explain our adaptation of the RePaint [32] technique mentioned in section 3, which was used for the partial layout conditioning examples in fig. 6, An overview is presented in algorithm 1. At every timestep, the intermediate sample $\mathbf{x}_i(t)$ is first updated with the rate of change provided by our model (v). Then the sample is slightly adjusted to conform to a path which will yield the values of the partial conditioning layout $\mathbf{y}_i$ at non-null entries after inference.

Some additional algorithm notation: The partial layout representation $\mathbf{y}_i = \{\mathbf{y}_i^j\}_{j \in J}$ is defined like the layout representation in section 3 extended by null values $\varnothing$, a placeholder value for entries of $\mathbf{y}_i$ tokens where no conditioning is provided. To give an example, consider a partial conditioning layout where the token $\mathbf{y}_i^j$ enforces that a bounding box with the label *chair* must be present in the final layout, but can have any coordinates. We set $\mathbf{b}_i^j = (\varnothing, \varnothing, \varnothing, \varnothing)$, $\mathbf{c}_i^j$ to be the PCA-reduced CLIP embedding of the word *chair* and $\alpha_i^j = 1$,

Figure 9: An example question page from our survey. Users must rate collections of 3 images from very implausible to very plausible. The underlying layout generators for the collections shown are (from top to bottom): No Layout, GPT4o, LayoutTransformer, and our method. Collection order was randomized for each question presented to the participant. Users click the button to select their rating.

and write:

$$\mathbf{y}_i^j = (\mathbf{b}_i^j \parallel \mathbf{c}_i^j \parallel \alpha_i^j)\,. \tag{7}$$

The mask variable M on line 5 of our algorithm tracks which values of $\mathbf{y}_i$ are null-values, and masks these values out during the update on line 12. To perform this masking, we define the arithmetic on $\varnothing$ as follows:

$$\begin{aligned} &\varnothing + a = \varnothing \text{ for } a \in \mathbb{R}, \\ &\varnothing * a = \varnothing \text{ for } a \in \mathbb{R} - \{0\}, \\ &\varnothing * 0 = 0. \end{aligned} \tag{8}$$

We construct the drift vector $\mathbf{d}_i$ which encodes the directional constraints. We begin by initializing $\mathbf{d}_i$ to 0 in all entries. Then, we add constraints. For example, if there is a constraint that bounding box j must be left of bounding box j', then

$$\mathbf{d}_i^j \leftarrow \mathbf{d}_i^j + (\lambda, 0, 0, 0 \parallel 0 \parallel 0)\,, \tag{9}$$

$$\mathbf{d}_i^{j'} \leftarrow \mathbf{d}_i^{j'} + (-\lambda, 0, 0, 0 \parallel 0 \parallel 0)\,, \tag{10}$$

where λ is a small constant.

In the special case when no conditioning is provided or directional constraints are provided ($\mathbf{y}_i \equiv \varnothing, \mathbf{d}_i = 0$), this algorithm is identical to the rectified flow inference presented in section 3 of our main paper.

Algorithm 1 Partially Conditioned Layout Generation

```
conditionedInference( P_i , y_i , d_i ):
T ← 1200
Δt ← 1/T
t ← 0
M ← 0 where y_i = ∅ otherwise 1   //Create a binary mask for the conditioning layout
x_i(0) ~ N(0, I)                  //Sample the starting noise
while t < 1 do
    dx_i(t)/dt ← v(x_i(t), t, P_i)   //Calculate the rate of change of x_i(t) at timestep t
    t ← t + Δt                       //Update timestep t
    x_i(t) ← x_i(t − Δt) + dx_i(t−Δt)/dt · Δt   //Calculate x_i(t) for the next timestep
    y_i(t) ← y_i · t + x_i(0) · (1 − t)        //Calculate conditioning update y_i(t)
    x_i(t) ← y_i(t) ⊙ M + x_i(t) ⊙ (1 − M)     //Update x_i(t) with conditioning in masked area
    x_i(t) ← x_i(t) + d_i                      //Apply drift for all given directional constraints
end while
Return x_i(1)
```

C Generated Layout Metrics

In the following, we introduce four metrics to assess the generated scenes layouts' plausibility and variety, and display their results alongside the models' generation times.

Object Numeracy. Our metric O_{Num} assesses whether generated layouts contain the objects at the expected frequencies. We sample across a collection of global prompts ($\{P_i\}$). The probability distribution for expected occurrences of the object-label ℓ in layouts generated from the global prompt P_i is written q_i^ℓ, and the probability distribution derived from ground-truth layouts is p_i^ℓ. Our metric is the normalized sum of KL-divergence between these two distributions:

$$O_{\text{Num}} := \frac{\sum_{i,\ell} KL(p_i^\ell || q_i^\ell)}{|\{P_i\}|} \tag{11}$$

where lower scores indicate that the model produces layouts with more plausible object numeracy. For display purposes, O_{Num} is scaled by 10^2 in table 2

Positional Likelihoods. We introduce $l_{\text{Pos}}^{(1)}$, and $l_{\text{Pos}}^{(2)}$ to measure how plausible the objects in a generated layout are arranged. Let m index all bounding boxes of object-label ℓ for prompt i. For each object-label ℓ, we obtain a distribution k_i^ℓ with KDE of the object's bounding box $(\mathbf{b}_i^\ell)_m$ in all layouts with global prompt i. We compute the average likelihood over all objects and all global prompts, to measure the *first-order positional likelihood*:

$$l_{\text{Pos}}^{(1)} = \frac{\sum_{i,\ell,m} k_i^\ell((\mathbf{b}_i^\ell)_m)}{|\{(\mathbf{b}_i^\ell)_m\}|}\,. \tag{12}$$

A higher value for $l_{\text{Pos}}^{(1)}$ means that object bounding boxes are placed in reasonable locations in the layout.

We also want to measure the likelihood of spatial relationships between objects. Let m^* index all bounding boxes of object-label ℓ'. For each object-label pair (ℓ, ℓ'), we obtain a distribution estimated with KDE $k_i^{\ell,\ell'}$ for the difference in the bounding box dimensions. We compute the average likelihood over all objects and all global prompts from our distributions to measure the *second order positional likelihood*:

$$l_{\text{Pos}}^{(2)} = \frac{\sum_{i,\ell\neq\ell',m,m^*} k_i^{\ell,\ell'}((\mathbf{b}_i^\ell)_m - (\mathbf{b}_i^{\ell'})_{m^*})}{|\{(\mathbf{b}_i^\ell)_m\}|(|\{(\mathbf{b}_i^\ell)_m\}|+1)/2}\,. \tag{13}$$

Model	Object Numeracy (↓)	Positional Variance (↑)	1st Order Positional Likelihood (↑)	2nd Order Positional Likelihood (↑)	mIoU (↑)	Time (s) (↓)
Ranni	3.83	<u>218</u>	2.10	0.56	0.04	214
LayoutGPT	3.76	134	<u>3.18</u>	0.81	0.06	81
GPT4o	3.71	93	4.17	<u>1.42</u>	<u>0.10</u>	111.0
LayoutDM	2.12	65	1.47	0.71	0.00	138.0
LayoutFlow	3.01	142	1.48	0.72	0.01	**0.5**
LayoutFlow (More steps)	2.96	143	1.44	0.65	0.01	<u>15.5</u>
LayoutTransformer	**0.90**	**231**	3.09	1.21	0.15	25.0
Ours	<u>1.14</u>	187	**4.76**	**1.93**	**0.17**	<u>15.5</u>

Table 2: **Layout Metrics, and Inference Speed.** A comparison of our metrics introduced in section 4. Our method achieves the best on mIoU, 1st and 2nd Order Positional Likelihood, while LayoutTransformer is highest Object Numeracy and Positional Variance. Closer inspections in table 1, fig. 4 reveal that LayoutTransformer falls short in terms of plausibility and image quality, indicating that it generates a large variety with plausible objects but physically implausible layouts.

A higher value for $l^{(2)}_{\text{Pos}}$ means that pairs of objects are plausibly positioned relative to one another. We conduct a grid search across bandwidths with 5-fold cross validation to optimize the KDE bandwidths for both $l^{(1)}_{\text{Pos}}$ and $l^{(2)}_{\text{Pos}}$.

Positional Variance. Our metric σ^2_{Pos} measures the variety of bounding boxes. For every bounding box $(\mathbf{b}^{\ell}_i)_m$, we find the bounding box in layouts with global prompt i and object label ℓ that is closest in Euclidean distance to the bounding box. We now redefine $\{m^*\}$ as the set of indices of bounding boxes in other samples which minimize the term $||(\mathbf{b}^{\ell}_i)_m - (\mathbf{b}^{\ell'}_i)_{m^*}||$. We compute all of these Euclidean distances and take the average:

$$\sigma^2_{\text{Pos}} = \frac{\sum_{i,\ell,m} \sum_{\{m^*\}} ||(\mathbf{b}^{\ell}_i)_m - (\mathbf{b}^{\ell'}_i)_{m^*}||}{\sum_{i,\ell,m} |\{m^*\}|} \tag{14}$$

If this metric is small, it means that the variance is low.

We provide results in table 2. We achieve the highest performance in positional likelihood scores and mIoU. While LayoutTransformer outperforms our model on *object numeracy* and *positional variance*, we observe that the layouts lack spatial plausibility (first and second order positional likelihood in table 2). This is also reflected in fig. 5: for example, the floor in the leftmost example appears at the top and the ashcan on the rightmost example is significantly too large. Our method ranks second in speed only to LayoutFlow, but we observe no definitive improvement in its layout statistics when the number of inference steps are raised to match our model's speed.

D Training Data and Hyperparameters

Our model consists of 20 AdaLN transformer blocks with 12-headed attention. For a token $\mathbf{x}^j$, we sinusoidally encode $\mathbf{b}^j$ into $\mathbb{R}^{72}$, and α^j into $\mathbb{R}^{18}$. $\mathbf{c}^j$ consists of the 30 top principal components of the object-label's CLIP embedding, which accounts for 77.35% of the explainable variance of our embeddings found in our training data. The timestep t is sinusoidally encoded into $\mathbb{R}^9$, while the CLIP embedding of a global prompt ℓ is down-projected by a trainable linear layer into $\mathbb{R}^{17}$ before interfacing with the AdaLN block.

When reporting model parameters, we include all transformer block weights and attached linear layers, including the PCA projection matrices. Given that CLIP dominates the number of parameters, it is a necessary subcomponent for InstanceDiffusion, and needed to form any complete text-to-image pipeline, we factor it out.

We train our model for 2000 epochs using stochastic gradient descent with learning rate $\lambda = 0.0005$ and a batch size of 32, using the Adam optimizer. We train on a Nvidia A100 GPU with 16 Intel Xeon Platinum Prozessor 8360Y CPUs with 244 GB RAM for approximately 20 hours. Baselines were trained according to their original training regimes on these same resources.

Due to limited compute, we did not have the resources to ablate these hyperparameters, and chose them as they yielded stable training and computational efficiency. In future work,

we hope to do so. Additionally, we evaluate on the full split of ADE20K, as spitting into evaluation, and then further splitting up into scene categories needed for evaluation, would leave very few samples left, causing concerns about stability. In future work, we hope to address this issue by scaling to larger datasets.

E Data and Code Access

We provide the code to our method, baselines, evaluations, and model weights at `https://huggingface.co/AnonymousSubmission42/SLayR`. Please download and unzip all files, and begin with the README.md in SLayr.zip.

F Limitations

One limitation of our work is that we do not currently scale up to large scale datasets such as MSCOCO [28] or LAION 5B [47] after it is passed through a layout annotation pipeline as in [52]. We did not scale up due both to lack of sufficient compute resources, and because our UI generation baselines CANNOT scale to an open set of captions. Therefore, to study the largest possible range of models, we focus primarily on this smaller dataset. For future work, we would like to investigate how the model scales up.

Another limitation is that SLayR does not directly produce text, rather a CLIP embedding which must then be mapped to text. However, this is standard practice in other vision fields such as 3D language fields ([19, 39]). In future work, we hope to experiment with text decoders to directly produce text.

As mentioned in appendix D, we did not have enough compute to conduct desired ablations on our hyperparameters. In future work, we hope to optimize the hyperparameter search space.

G Broader Impacts

We acknowledge that research towards text-to-image generative AI can be misused for the purposes of deep fakes or plagiarism of artistic content.

H Safeguards

We have trained our model exclusively on publicly available and curated datasets to mitigate the risk of generating inappropriate content.

In our code README, we also implore users to refrain from using our model for deep fake generation.

I Licenses

Models:

- LayoutTransformer [12]: `https://github.com/kampta/DeepLayout`, Apache 2.0 `https://www.apache.org/licenses/LICENSE-2.0`
- LayoutFlow [11]: `https://github.com/JulianGuerreiro/LayoutFlow`, MIT `https://opensource.org/license/mit`
- LayoutDM [16]: `https://github.com/CyberAgentAILab/layout-dm` Apache 2.0 `https://www.apache.org/licenses/LICENSE-2.0`
- Ranni [8]: `https://github.com/ali-vilab/Ranni`, Apache 2.0 `https://www.apache.org/licenses/LICENSE-2.0`

- LLM-Blueprint [9], https://github.com/hananshafi/llmblueprint, no license could be found, however use of the repo, with proper citation, is encouraged in README.md.
- LayoutGPT [7]: https://github.com/weixi-feng/LayoutGPT, MIT https://opensource.org/license/mit
- LLM-GroundedDiffusion [26]: https://github.com/TonyLianLong/LLM-groundedDiffusion, no license could be found, however use of the repo, with proper citation, is encouraged in README.md.

Metrics:

- CMMD [17]: https://github.com/sayakpaul/cmmd-pytorch, Apache 2.0 https://www.apache.org/licenses/LICENSE-2.0
- FID [13]: https://github.com/Lightning-AI/torchmetrics/blob/master/src/torchmetrics/image/inception.py, Apache 2.0 https://www.apache.org/licenses/LICENSE-2.0
- KID [3]: https://github.com/Lightning-AI/torchmetrics/blob/master/src/torchmetrics/image/kid.py, Apache 2.0 https://www.apache.org/licenses/LICENSE-2.0
- VQA [29]: https://github.com/linzhiqiu/t2v_metrics, Apache 2.0 https://www.apache.org/licenses/LICENSE-2.0
- HPSv2 [54] : https://github.com/tgxs002/HPSv2, Apache 2.0 https://www.apache.org/licenses/LICENSE-2.0
- ImageReward [58]: https://github.com/zai-org/ImageReward, Apache 2.0 https://www.apache.org/licenses/LICENSE-2.0

Packages:

- matplotlib: BSD https://github.com/nilearn/nilearn/blob/main/LICENSE
- pytorch: https://github.com/pytorch/pytorch/blob/main/LICENSE

Datasets:

- ADE20K[65]: https://ade20k.csail.mit.edu/ BSD-3 https://opensource.org/license/BSD-3-Clause

J LayoutTransformer Temperature

Throughout our main paper, we maintained LayoutTransformer defaulttemperature parameter equal to one. However, the question arises whether the generated layouts would be higher quality at lower temperatures, where the model's output is more stable. As shown in table 3 even when we select the lowest temperature of zero for optimal stability, we are still not measuring a decisive improvement across numerical metrics, therefore we kept the temperature at its original setting of one to remain as faithful as possible to the prior work.

Model	**FID** (↓)	**KID** (10^{-2})(↓)	**CMMD** (↓)	O_{Num}(↓)	$l^{(1)}_{\text{Pos}}(10^{-11})$(↑)	$l^{(2)}_{\text{Pos}}(10^{-11})$(↑)	σ^2_{Pos}(↑)
LayoutTransformer temp= 1	<u>0.44</u>	0.94	<u>1.34</u>	**0.90**	3.09	1.21	**231**
LayoutTransformer temp= 0	0.48	<u>0.92</u>	1.77	4.11	<u>3.73</u>	<u>1.53</u>	0
Ours	**0.17**	**0.27**	**0.03**	<u>1.14</u>	**4.76**	**2.03**	187

Table 3: Comparison of metrics LayoutTransformer with a temperature of one (model default) and a temperature of zero. Even when the temperature is zero, we see that our method still performs better across our metrics.

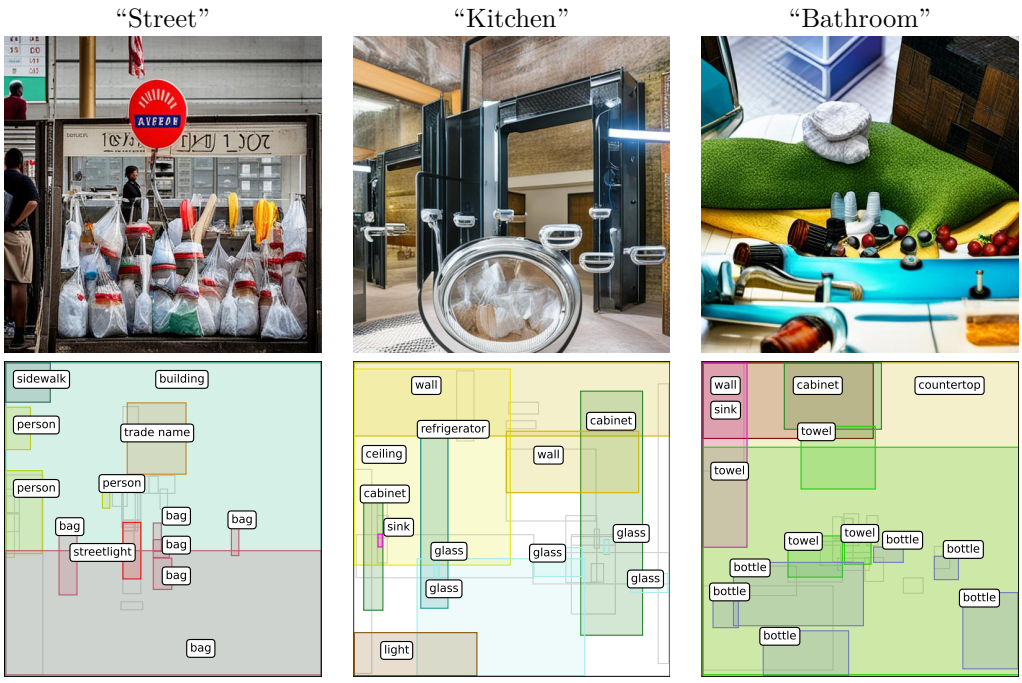


Figure 10: Example layouts and images for LayoutTransformer when temp=0. Even at the most stable setting, the images appear implausible. Objects that are typically small details, such as *bag*, *glass*, or *bottle* repeated many times across the layout.

K GPT4o Temperature

Because we observed low image variance for GPT4o layouts, we also considered what would happen if we raised the temperature of GPT4o from the default 0.25 as set in LLM-grounded Diffusion to achieve more variety.

We experimented with increasing the temperature from 0.2 in increments of 0.1. We found that at a temperature of 1, GPT4o failed to produce a parsable layout 14% of the time. However, these mistakes were easy to catch and query the model again. Temperatures higher than 1 caused more frequent parsing failures, and began to produce long, tangential sentences rather than proper object labels. Without a method to heuristically filter these responses, we settled on a temperature of one as a reasonable upper limit for operation temperature of GPT4o on this task.

We compare the performance of GPT4o with a temperature of one with our method, and GPT4o with the default temperature in table 4. Our model still outperforms GPT4o when the temperature is one in FID and KID. While raising the temperature improves the object numeracy score O_{Num} and the positional variance score σ^2_{Pos} improve in GPT4o when the temperature is raised, they are still worse than our method, and come at the cost of decreased performance in the positional likelihood scores $l^{(1)}_{\text{Pos}}$ and $l^{(2)}_{\text{Pos}}$. Therefore, raising the temperature does not offer a clear advantage on our numerical metrics.

We also visualized outputs of GPT4o with the raised temperature in fig. 11. Although there is some increase in the variation of scenes, the effect does not appear to be noticeably pronounced. Therefore, we choose to stick with a temperature of 0.25 for our human evaluation, as this is the most faithful adaptation of our LLM-grounded Diffusion baseline, without neglecting a clear optimization.

Model	FID (↓)	KID (10^{-2})(↓)	CMMD (↓)	O_{Num}(↓)	$l^{(1)}_{\text{Pos}}(10^{-11})$(↑)	$l^{(2)}_{\text{Pos}}(10^{-11})$(↑)	σ^2_{Pos}(↑)
GPT4o temp=0.25	0.94	0.99	1.34	3.71	4.37	1.49	93
GPT4o temp=1	1.47	1.62	1.35	2.86	4.02	1.35	142
Ours	**0.17**	**0.27**	**0.03**	**1.14**	**4.76**	**2.03**	**187**

Table 4: Comparison of metrics GPT4o with a temperature of 0.25 (adapted model default) and one (highest stable temperature). At increased temperatures, GPT4o performs worse on the FID and KID metrics. Although increasing the temperature of GPT4o improves O_{Num} (the object frequencies are closer to the ground truth) and σ^2_{Pos} (the layouts are more varied overall), performance on $l^{(1)}_{\text{Pos}}$ and $l^{(2)}_{\text{Pos}}$ drops (the positions of the objects are less plausible). Our method still performs better in all displayed metrics.

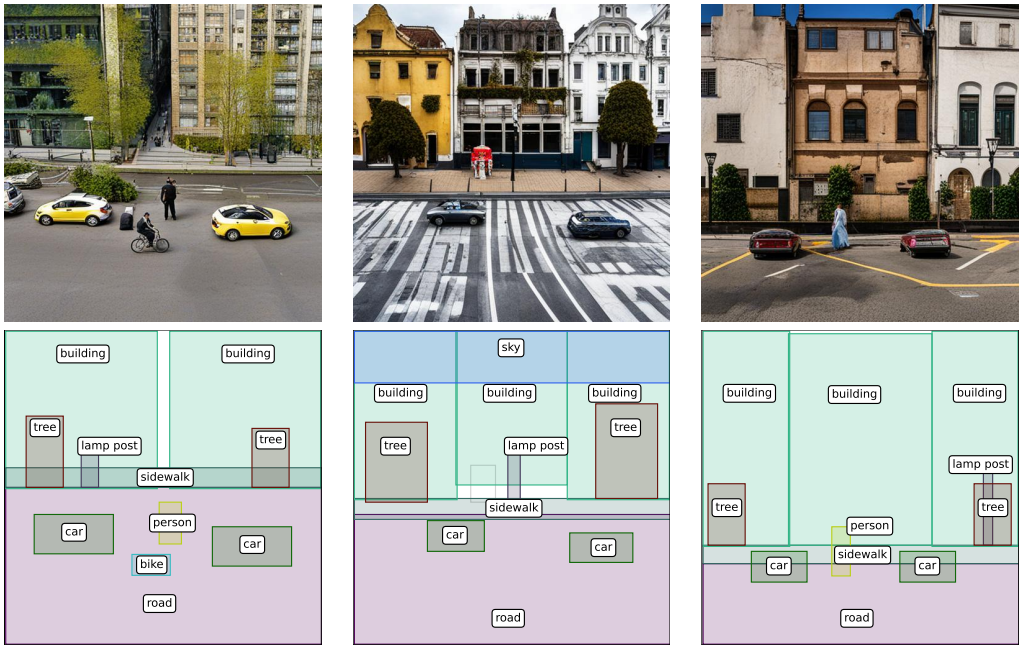


Figure 11: Example images and underlying layouts of the prompt *street* for GPT4o when the temperature is one, the highest stable temperature. Visually, there is slightly more variation than at a temperature of 0.25, (see fig. 24), but this is not a pronounced effect: positions and quantities of objects, and implied camera angle, are still very repetitive.

L GPT4o Query Template

We adapted the prompt template from LLM-grounded Diffusion by replacing the old scene caption and layout examples with ground truth data from ADE20k and encourage chain of thought reasoning [53]. Our LLM prompt is listed in fig. 12. Our in-context examples are listed in table 5.

```
Task Description and Rules
You are a smart program for automatic image layout generation.  I provide you
          with a global prompt which describes the entire image.  The image
          layout has a height of 512 and a width of 512.  The coordinate system
          assumes the origin (0,0) is in the top left corner.  Bounding box
          coordinates are specified in the format (x,y,w,h), where x and y are
          the top left corner coordinate, and w and h are the full width and
          height of the box.  Your task is to imagine which objects reasonably
          belong in an image with a global prompt, and arrange these objects in
          into a layout which could plausibly be for a real image.

Meta Command
Reason about the objects added to the layout For each object reason about its
          position in the layout relative to the other objects, and why it is
          likely.  In general maintain a plausible configuration of the objects
          within the image layout such that the coordinates obey our coordinate
          convention.  Do not number the objects, instead put them in a list
          in the exact format shown below.  Remember to include the caption,
          background prompt and negative prompt in the layout.

[ In - context Examples ]

Question
Provide the layout for a "{prompt}"
```

Figure 12: Our full prompt to the LLM for layout generation. LLM starts completions from "Objects."

```
Provide a layout for an "airport terminal"
Answer:
Caption:  An airport terminal
Objects:  [('ceiling',[1, 0, 510, 292]), ('floor',[0,
          360, 468, 151]), ('wall',[0, 337, 152, 152]),
          ('wall',[3, 193, 182, 100]),('first floor',[353,
          262, 158, 96]),('wall',[353, 207, 158, 86]),('first
          floor',[0, 264, 136, 93]),('plant',[177, 398, 158,
          78]),('fountain',[197, 376, 122, 79]),('first
          floor',[134, 310, 265, 30]),('wall',[398, 343, 68,
          117]),('wall',[467, 330, 44, 172]),('wall',[190,
          270, 157, 42]),('column',[65, 159, 30,
          157]),('column',[0, 76, 19, 242]),('column',[442, 153,
          30, 141]),('wall',[275, 336, 123, 33]),('wall',[134,
          340, 129, 31]),('fence',[331, 394, 44, 83]),('first
          floor',[110, 282, 86, 39]),('column',[404, 171, 23,
          139]),('wall',[239, 499, 272, 12]),('column',[113,
          175, 22, 144]),('seat',[187, 478, 134,
          17]),('column',[380, 193, 19, 117]),('fence',[134,
          403, 40, 53]),('column',[145, 196, 18,
          116]),('tree',[345, 324, 34, 45]),('tree',[43, 404,
          30, 49]),('tree',[150, 329, 32, 45])]
Background prompt:  an airport terminal
Negative prompt:  empty
Reasoning:  Airport terminals contain many walls and columns,
          and have a floor and ceiling.  They also contain
          seats for passengers to wait in as well as decorative
          trees

Provide a layout for an "schoolhouse"
Answer:
Caption:  schoolhouse
Objects:  [('sky',[0, 0, 510, 431]),('building',[22, 23, 460,
          465]),('tree',[1, 0, 173, 200]),('grass',[0, 449, 510,
          61]),('tree',[422, 129, 83, 281]),('tree',[1, 202,
          44, 254]),('path',[0, 478, 308, 27]),('grass',[0, 469,
          201, 27]),('plant',[42, 412, 71, 76]),('plant',[399,
          417, 56, 69]),('person',[229, 412, 33, 64]),('car',[3,
          444, 40, 29]),('tree',[0, 431, 41, 23]),('tree',[472,
          426, 37, 23])]
Background prompt:  schoolhouse
Negative prompt:  empty
Reasoning:  A schoolhouse is typically a building.  The layout
          could include a path, students, trees, plants, and a
          car in the schoolyard.

Provide a layout for an "ball pit"
Answer:
Caption:  ball pit
Objects:  [('inflatable park',[1, 0, 510, 510]),('person',[85,
          42, 313, 398]),('ball',[451, 292, 48,
          69]),('ball',[77, 253, 46, 61]),('ball',[416, 278,
          40, 58]),('ball',[475, 265, 34, 68]),('ball',[371,
          240, 39, 55]),('ball',[430, 246, 40, 47])]
Background prompt:  ball pit
Negative prompt:  empty
```

```
Reasoning: A ball pit is an inflatable park with balls and
           people. The layout could include a person playing in
           the ball pit and colorful balls scattered around the
           inflatable park.

Provide a layout for an "jail cell"
Answer:
Caption: jail cell
Objects: [('bar',[0, 0, 510, 512]),('floor',[24, 304, 390,
          206]),('wall',[296, 16, 156, 482]),('wall',[72, 4,
          232, 302]),('bed',[174, 256, 234, 196]),('cell',[462,
          26, 48, 484]),('wall',[20, 4, 50, 458]),('shelf',[66,
          48, 242, 20]),('sink',[152, 194, 40, 54])]
Background prompt: jail cell
Negative prompt: empty
Reasoning: A jail cell typically has bars, walls, a floor, and
           a bed. The layout could include a cell door, a shelf,
           and a sink.

Provide a layout for an "badlands"
Answer:
Caption: badlands
Objects: [('earth',[0, 199, 334, 267]),('earth',[58, 201,
          453, 144]),('hill',[0, 106, 512, 118]),('sky',[0,
          0, 512, 116]),('earth',[194, 334, 316,
          177]),('water',[34, 218, 301, 128]),('tree',[0, 369,
          236, 142]),('rock',[0, 381, 97, 83]),('person',[463,
          273, 18, 71]),('tripod',[450, 289, 5, 38]),('photo
          machine',[449, 283, 8, 8])]
Background prompt: badlands
Negative prompt: empty
Reasoning: Badlands are characterized by eroded rock
           formations, so the layout could include earth, hills,
           rocks, and trees. The badlands may also have water,
           a person, a tripod, and a photo machine.

Provide a layout for an "art gallery"
Answer:
Caption: art gallery
Objects: [('wall',[224, 36, 287, 360]),('floor',[0, 323, 512,
          188]),('wall',[0, 84, 226, 261]),('ceiling',[0, 0,
          511, 112]),('board',[306, 153, 205, 140]),('board',[0,
          170, 250, 102]),('double door',[251, 176, 55,
          168]),('grill',[378, 260, 21, 91]),('grill',[338,
          257, 20, 85]),('vent',[248, 22, 47,
          19]),('drawing',[490, 196, 21, 40]),('spotlight',[453,
          32, 17, 49]),('drawing',[8, 194, 18,
          35]),('spotlight',[381, 54, 14, 43]),('drawing',[456,
          241, 22, 25]),('spotlight',[279, 83, 15,
          35]),('spotlight',[320, 71, 14, 38]),('drawing',[391,
          187, 17, 30]),('drawing',[314, 201, 20,
          26]),('vent',[6, 45, 36, 13]),('spotlight',[259,
          88, 12, 34]),('drawing',[420, 196, 14,
          28]),('drawing',[445, 204, 18, 22]),('drawing',[409,
          239, 13, 28]),('spotlight',[234, 97, 11,
          32]),('drawing',[43, 195, 18, 18]),('drawing',[351,
          181, 12, 27]),('drawing',[135, 237, 17,
          19]),('drawing',[40, 227, 14, 23]),('drawing',[205,
          200, 11, 27])]
```

```
Background prompt:  art gallery
Negative prompt:  empty
Reasoning:  An art gallery is indoors, so it has walls, a
            floor, and a ceiling.  It can also have boards for
            displaying art, doors, grills, vents, and spotlights.
            The art gallery may have drawings on the walls and
            spotlights to illuminate the art.
Provide a layout for an "art gallery"
Answer:
Caption:  window seat
Objects:  [('seat',[2, 172, 507, 337]),('floor',[28, 322,
           482, 187]),('wall',[102, 0, 266, 228]),('wall',[0,
           0, 109, 510]),('person',[222, 20, 133,
           390]),('wall',[363, 0, 146, 324]),('windowpane',[140,
           0, 204, 69]),('windowpane',[0, 0, 102,
           122]),('windowpane',[388, 0, 122, 75]),('hat',[375,
           157, 80, 69])]
Background prompt:  window seat
Negative prompt:  empty
Reasoning:  A window seat typically has a seat, walls, and a
            floor.  The layout could include a person sitting on
            the seat, looking out the window, and wearing a hat.
```

Table 5: Our in-context examples. We use fixed in-context examples for layout generation.

M Comparison to DDIM

We initially considered a DDIM [50] based approach rather than rectified flow. However, early experiments showed less promise in this direction. DDIM models struggled with generating the correct CLIP embeddings, leading to meaningless images that did not match the prompt, whereas rectified flow-based approaches were more successful without needing to search the hyperparameter space.

We provide an example here, from a model with an identical architecture to our presented model (including all hyperparameters specified in appendix D, except it is trained with a DDIM training objective and performs DDIM inference (with a log-linear noise schedule from $\sigma = 0.02$ to $\sigma = 1$). This is not an exhaustive search by any means, but is intended as a point-of-reference for other researchers.

We show our statistics in table 6, and some visual examples from the model in fig. 13. We speculate that the straighter transit paths of samples rectified flow [30] increases the model's ability to effectively learn high dimensional data like the PCA-reduced CLIP embeddings.

Model	**FID** (↓)	**KID** (10^{-2})(↓)	**CMMD** (↓)	O_{Num}(↓)	$l^{(1)}_{\text{Pos}}$(↑)	$l^{(2)}_{\text{Pos}}$(↑)	σ^2_{Pos}(↑)
Ours (DDIM)	0.95	8.60	1.77	7.89	4.33	0.01	**239**
Ours (Rectified Flow)	**0.17**	**0.27**	**0.03**	**1.91**	**4.76**	**2.03**	187

Table 6: Generated image metrics, and our generated layout numerical metrics applied on our model architecture with DDIM or rectified flow. Our model performs better on everything except positional variance σ^2_{Pos}, but this is at the cost of the layouts being largely nonsense (see fig. 13)

N Additional Images and Layouts

Here we present additional examples of our model's generated layouts, and conditionally generated images, for the prompts *bedroom* (fig. 14), *mountain* (fig. 15), and *kitchen* (fig. 16).

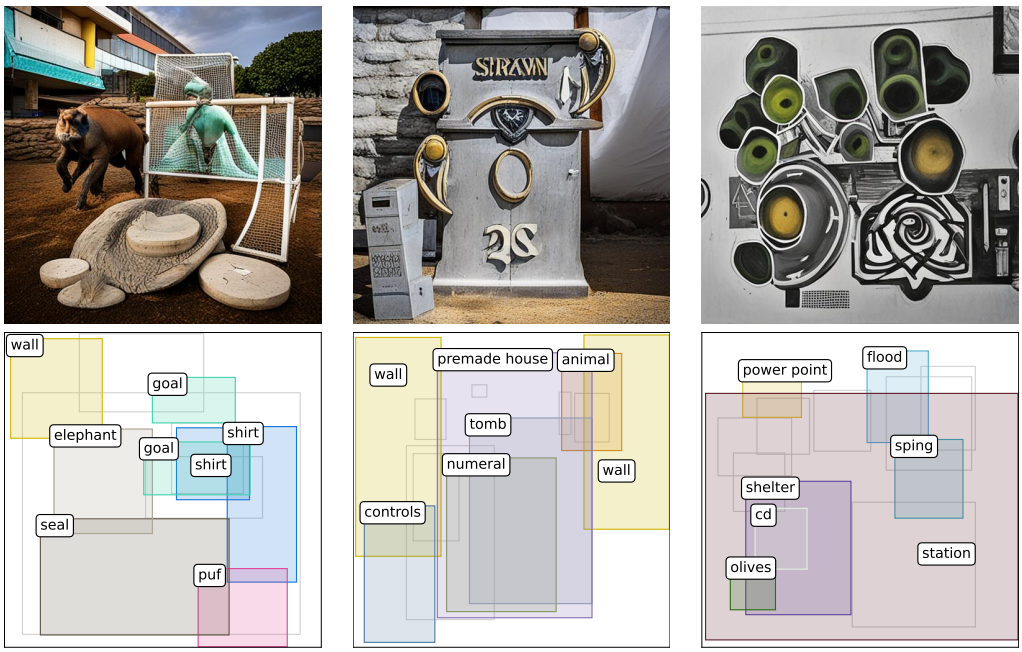


Figure 13: Our Model with DDIM instead of Rectified Flow - Street. The bounding box labels match poorly to the desired scene, and the resulting images appear to be implausible.

O Editing Capabilities: Comparison with Drag-based Manipulation Methods

In fig. 7, we show how a pipeline using our model supports image editing functionalities like relocating or removing objects. Here, we compare these capabilities against Readout Guidance [33], which enables users to move visual elements via guidance arrows.

As shown in fig. 17, Readout Guidance fails to relocate the plant to the floor when instructed, wheras our method suceeds. We also try to fully remove objects with Readout Guidance by dragging them to the far edge of the image. In this case, the former plant location is replaced with a black patch, not a realistic inpainting.

Results in Readout Guidance are primarily for small transformations, and our case study suggests it might struggle with longer range manipulations. Thus, text-to-layout-to-image approach with explicit layout-based explicit control can be a more attractive approach to editing generated images, as it seems to perform more strongly.

P Selecting the Opacity Threshold 0.5

We visualize the distribution of generated $\alpha_i^j(1)$ of our model on the ADE20K benchmark in fig. 18. The values cluster around 0 and 1, meaning the model makes a strong distinction between which tokens should be recognized or ignored in a scene layout. We select $\alpha_i^j(1) < 0.5$ as a unbiased threshold.

Q Print-ready Main Results Diagram

For readers who prefer the document on paper, we include our visual results diagram from fig. 5 at a size where the annotations are large enough to be printed clearly. The annotations can also be zoomed into on our main document.

Figure 14: Ours - Bedroom.

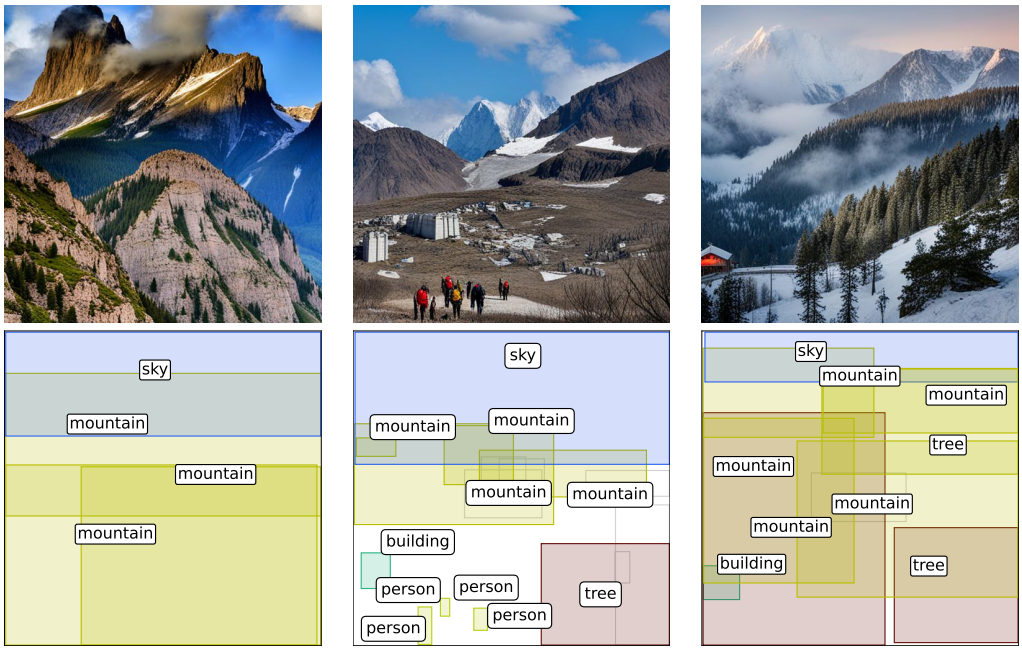


Figure 15: Ours - Mountain.

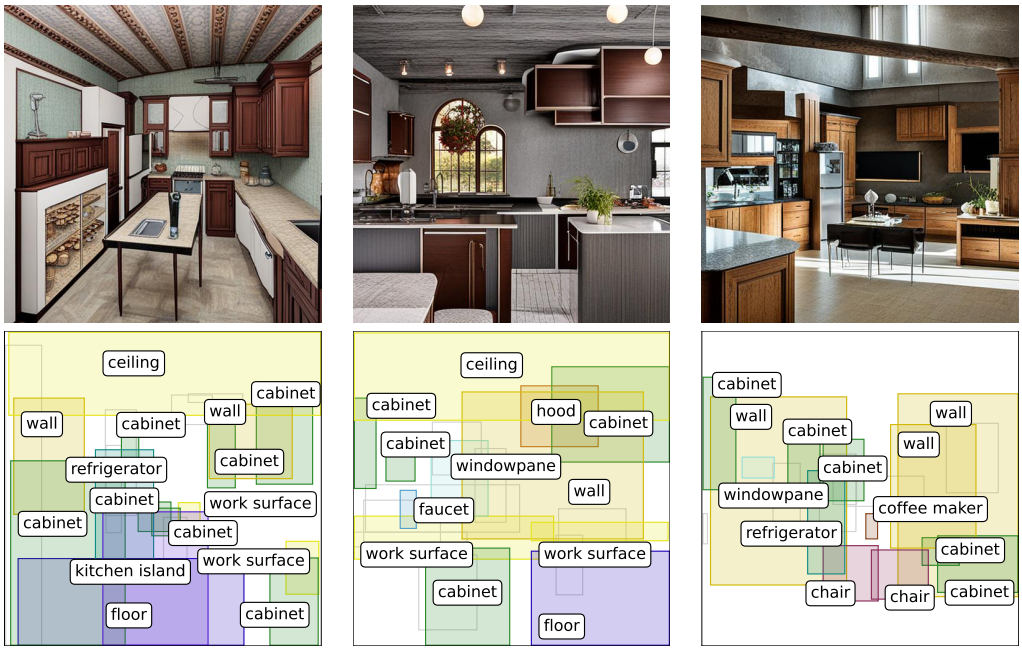


Figure 16: Ours - Kitchen.

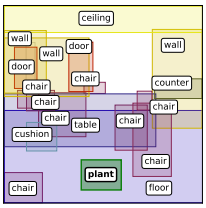
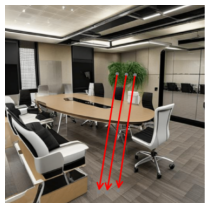
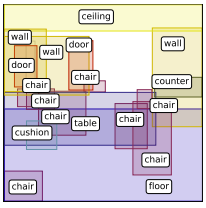
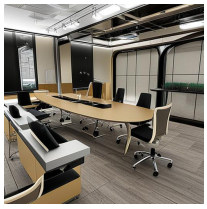
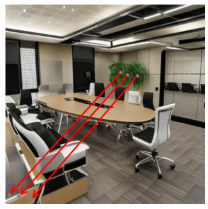
	Edited Layout	Our Generated Image	Arrow Diagram	Generated Image by Readout Guidance
Image Relocation				
Image Removal				

Figure 17: Comparison of editing abilities. Each row corresponds to an editing task: **Top:** relocating the plant, **Bottom:** removing the plant.

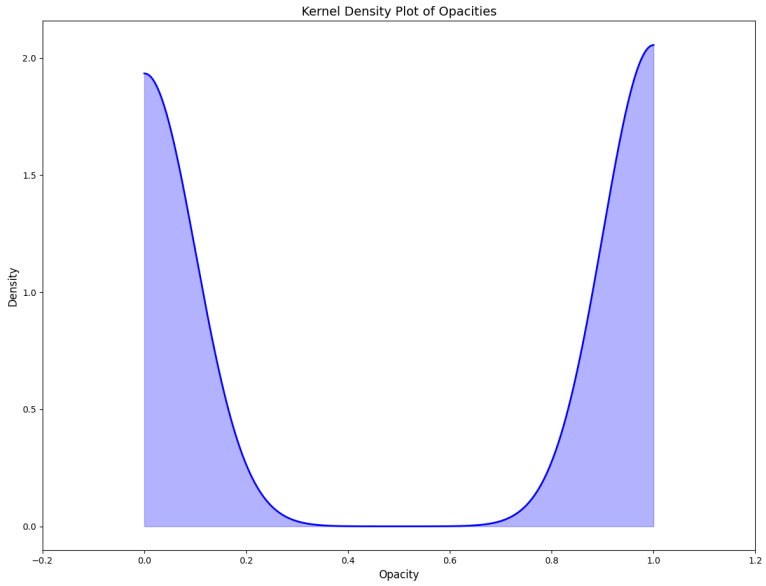


Figure 18: **Opacity KDE** Opacities generated by our model cluster towards 0 and 1, the ground truth opacities shown during training.

Figure 19: No Layout - Living Room.

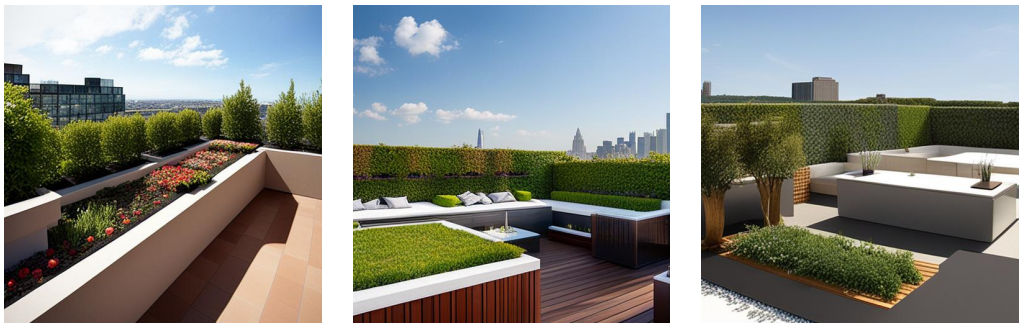


Figure 20: No Layout - Roof Top

Figure 21: No Layout - Street

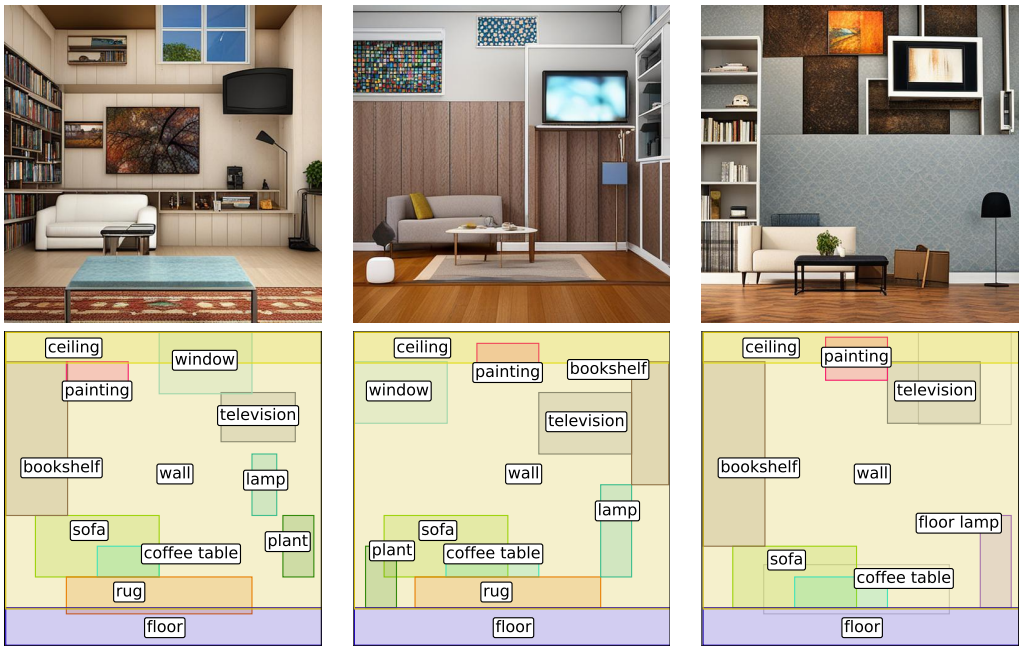


Figure 22: GPT4o - Living Room.

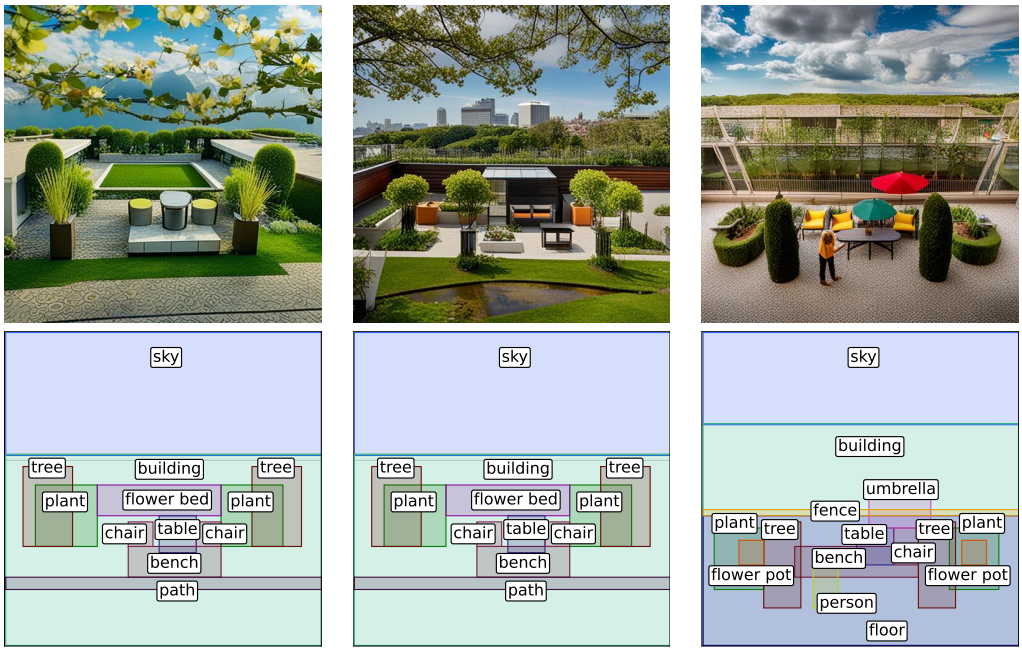


Figure 23: GPT4o - Roof Top.

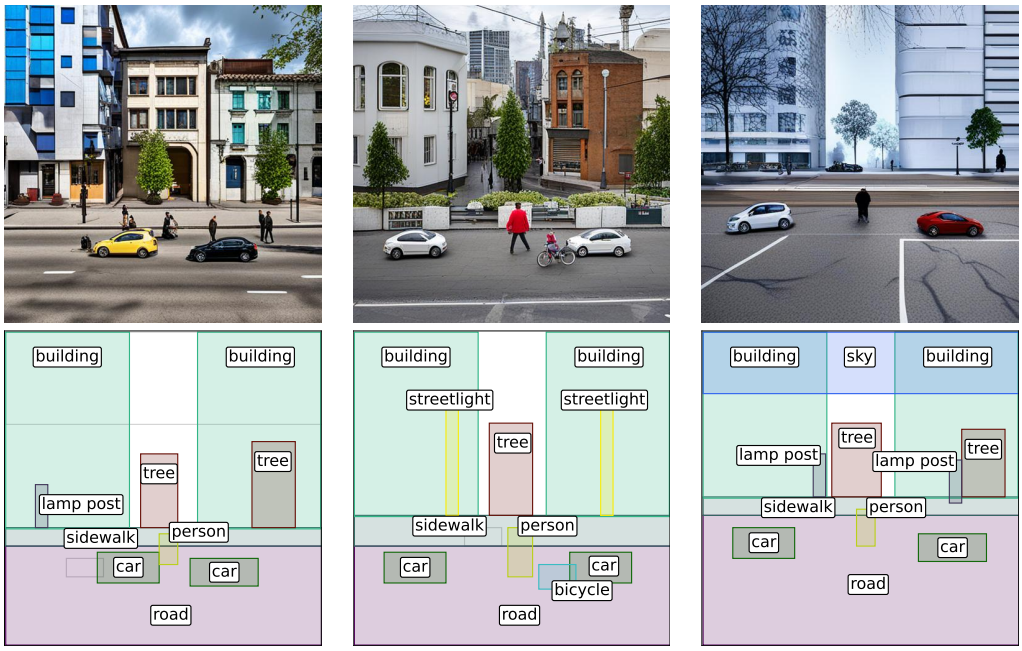


Figure 24: GPT4o - Street.

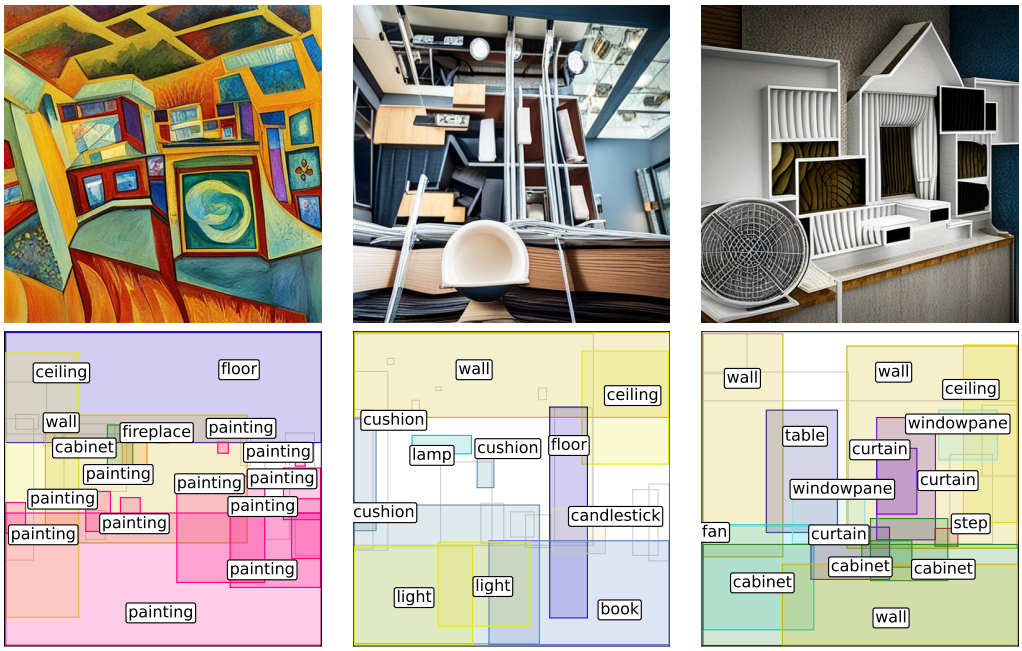


Figure 25: Layout Transformer - Living Room.

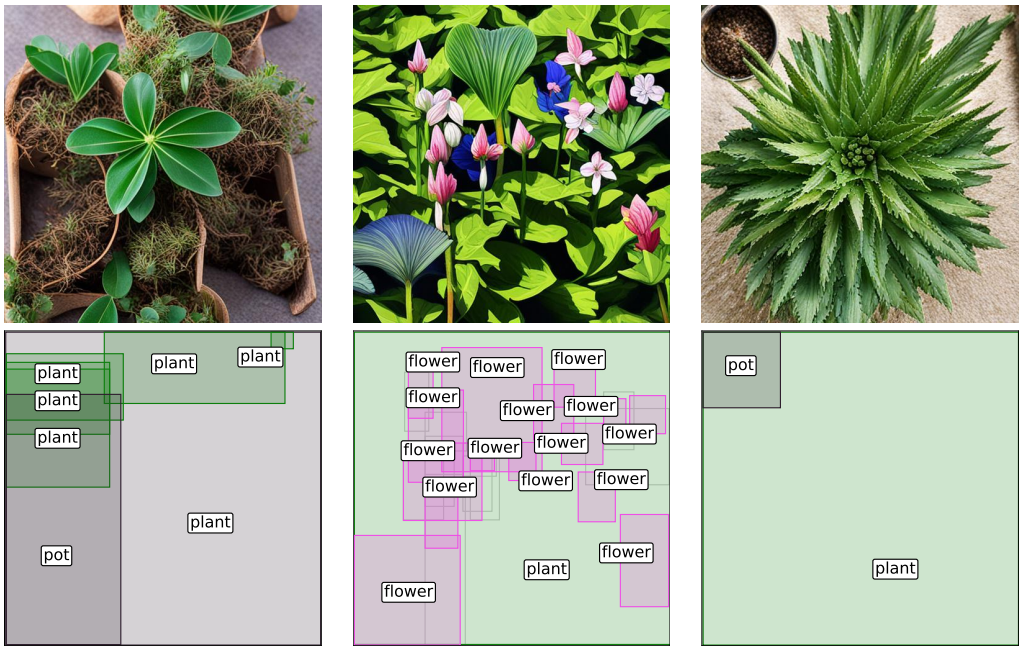


Figure 26: Layout Transformer - Roof Top.

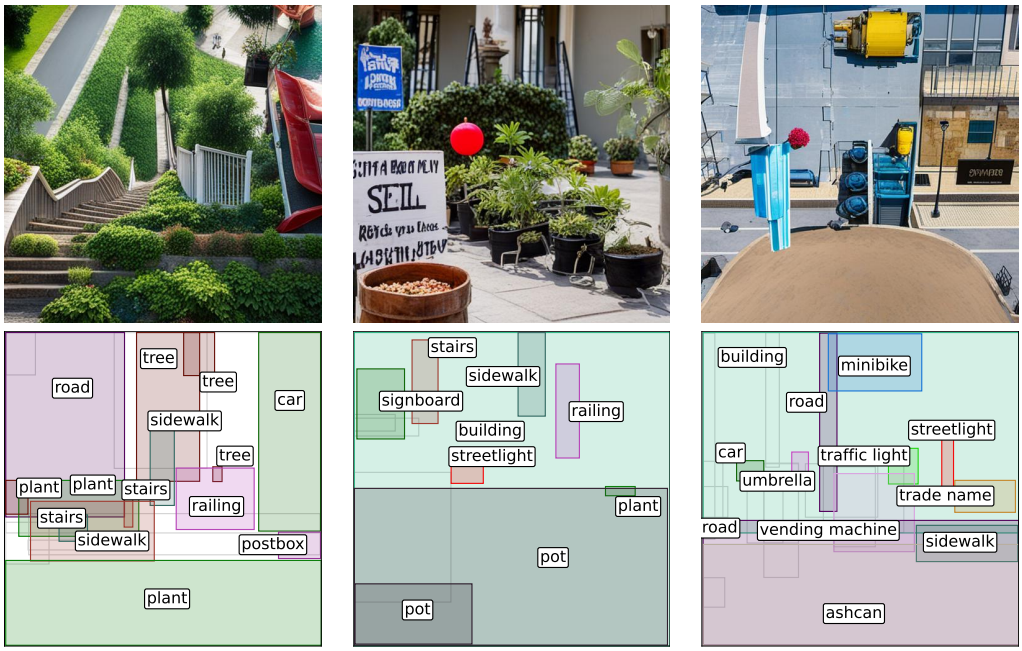


Figure 27: Layout Transformer - Street.

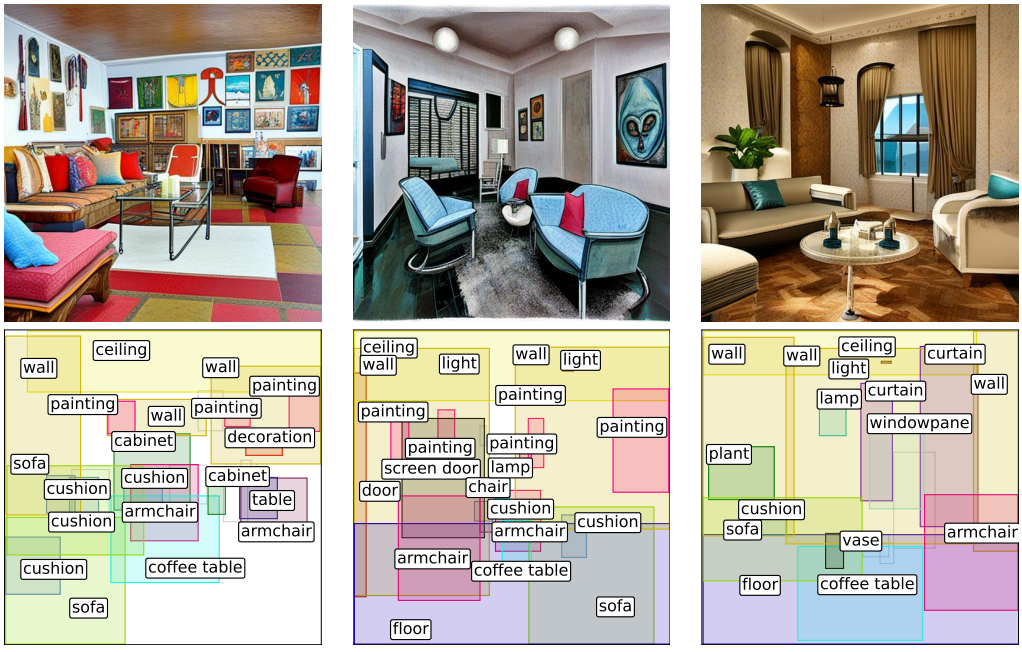


Figure 28: Ours - Living Room.

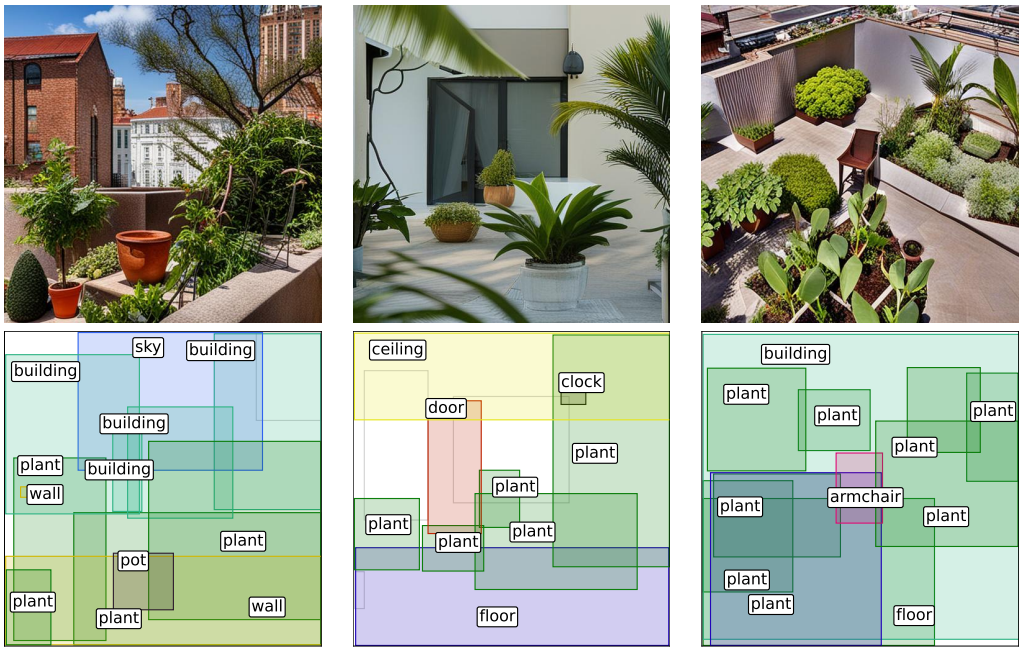


Figure 29: Ours - Roof Top.

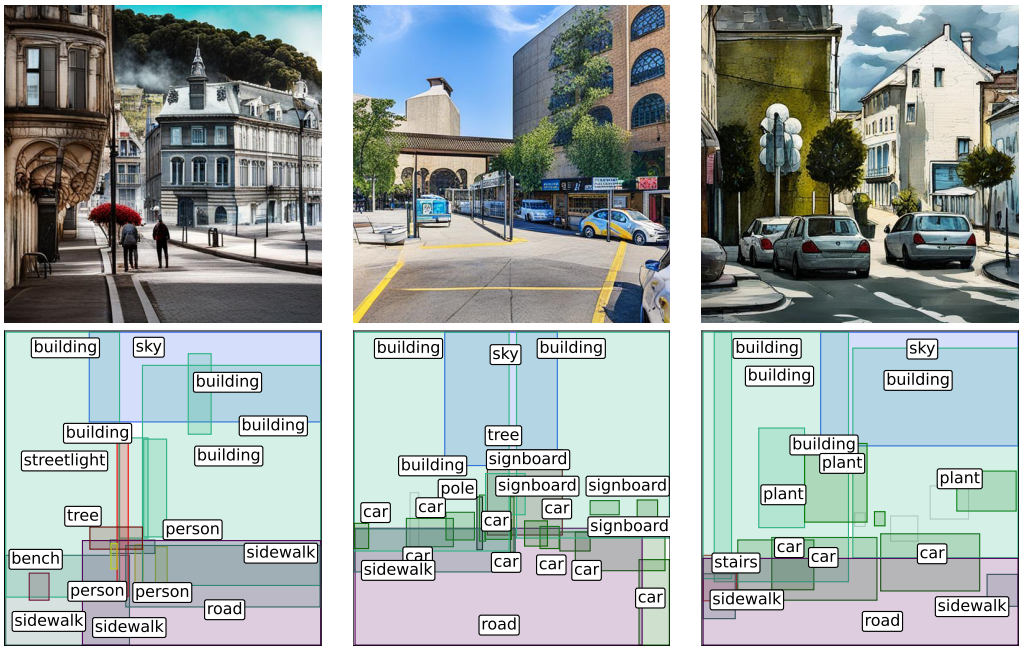


Figure 30: Ours - Street.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: At the end of our introduction we supply our claims (**1)** we introduce the first model for rectified flow-based text-to-layout generation and show that it produces a large variety of highly plausible layouts for challenging unconstrained prompts, **2)** we establish a well-designed human-evaluation study that can be repeated by others, and **3)** demonstrate that integrating our method into a complete text-to-layout-to-image pipeline yields state-of-the-art in achieving variety and plausibility together. See our supplement to access source code.), which we support through our findings in the paper

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations in appendix F

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The

authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: Our method does not provide theoretical results.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Our method section section 3 describes our architecture, and we provide hyperparameters in appendix D. Although we cover the major details there, any details we might have missed can be ascertained from our code, provided in appendix E. We provide details on our human survey method in appendix A.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.

(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: In appendix E, we provide a link to our data and code.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide information on our data in section 4 and information and explanations of our optimizer and hyperparameters in appendix D. We provide details on our human survey method in appendix A.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: In section 4, we explain our standard error bars shown in fig. 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.

- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We describe inference compute resources in section 4 (see speed measurement explanations), display the model parameter counts in fig. 4, and explain training resources in appendix D.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the code and found no violations. Furthermore, we explain the ethics of our crowd sourced experiments in checklist items below.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss broader impacts in appendix G

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: We discuss safe guards in appendix H

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We discuss licenses in appendix I

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We provide a link to our source code repository in appendix E. There there is further documentation.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [Yes] .

    Justification: In appendix A, we include the instruction text given to the participants, as well as a screenshot from a sample survey page. We also provide details on the participants' compensation, which obeys local wage regulations.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [Yes] .

    Justification: As written in appendix A, the study was approved by our Institutional Review Board (referred to as the Ethics Review Board of our institution within the text). We also could not see any risks to the participants from participation in our survey, and therefore had nothing clear to disclose. However, we still informed participants with a summary of the survey and asked for their informed consent before they proceeded, in order to best ensure participant safety.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA] .

    Justification: LLMs were not used for any of the core methods or writing of this paper.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.