Generative Floor Plan Design with LLMs via Reinforcement Learning with Verifiable Rewards

Luis Lara[1], Aristides Milios[1,2], Zhi Hao Luo[1,3], Aditya Sharma[1,3], Ge Ya Luo[1,2], Christopher Beckham[1], Florian Golemo[1], Christopher Pal[1,2,3,4]
[1]Mila – Quebec AI Institute, [2]Université de Montréal,
[3]Polytechnique Montréal, [4]Canada CIFAR AI Chair
Correspondence: luis.lara@mila.quebec

Abstract

An AI system for professional floor plan design must precisely control room dimensions and areas while respecting the desired connectivity between rooms and maintaining functional and aesthetic quality. Existing generative approaches focus primarily on respecting the requested connectivity between rooms, but do not support generating floor plans that respect numerical constraints. We introduce a text-based floor plan generation approach that fine-tunes a large language model (LLM) on real plans and then applies reinforcement learning with verifiable rewards (RLVR) to improve adherence to topological and numerical constraints while discouraging invalid or overlapping outputs. Furthermore, we design a set of constraint adherence metrics to systematically measure how generated floor plans align with user-defined constraints. Our model generates floor plans that satisfy user-defined connectivity and numerical constraints and outperforms existing methods on Realism, Compatibility, and Diversity metrics. Across all tasks, our approach achieves at least a 94% relative reduction in Compatibility compared with existing methods.[1] Our results demonstrate that LLMs can effectively handle constraints in this setting, suggesting broader applications for text-based generative modeling.

1 Introduction

Generative models have emerged as powerful tools for accelerating design across different sectors. However, their widespread adoption depends on their ability to strike a balance between flexibility and precise control. This challenge is particularly pronounced in floor plan generation, where users often need to specify strict constraints, including exact room sizing and connectivity. To address this, we propose a model that handles user-defined constraints, and we introduce evaluation metrics that measure how well generated floor plans satisfy those constraints.

Most existing generative modeling techniques for floor plan design rely on top-down 2D renderings and use input graphs, referred to as bubble diagrams, to represent the spatial connectivity between rooms. However, such approaches have inherent drawbacks. Vision-based generative models, for instance, produce rasterized image-based outputs, making it challenging to access or modify them directly. Additionally, many models focus exclusively on adjacency constraints, disregarding room geometry entirely, thereby reducing the degree of control users can exert over the generated layouts.

To overcome these limitations, we adopt a JSON-based representation for generative floor plan modeling that encodes room layouts as polygonal structures. Our method builds on prior vectorized floor plan approaches, such as *House-GAN*, *House-GAN++* (Nauata et al., 2020, 2021), and *HouseDiffusion* (Shabani et al., 2023), and leverages data from the *RPLAN* dataset (Wu et al., 2019). This structured format facilitates greater control over spatial parameters, including room size and connectivity. To validate this approach, we fine-tune a large language model (LLM) to generate JSON-encoded floor plans with spatial constraints specified in the prompt. See Figure 1 for an overview of the inference process.

Through our experiments, we validate the effectiveness of our approach in maintaining high compliance with the specified constraints, as measured by our proposed constraint-adherence metrics. In the most complex setting, the eight-room task, our approach with best-of-10 sampling reduces Compatibility by 94% relative to *HouseDiffusion* (Table 1). Compatibility is a standard metric of how well the generated floor plan matches the desired room connectivity, where lower is better.

[1]Project code is available at https://github.com/ludolara/floor-plan-rlvr

Findings of the Association for Computational Linguistics: ACL 2026, pages 26612–26627
July 2-7, 2026

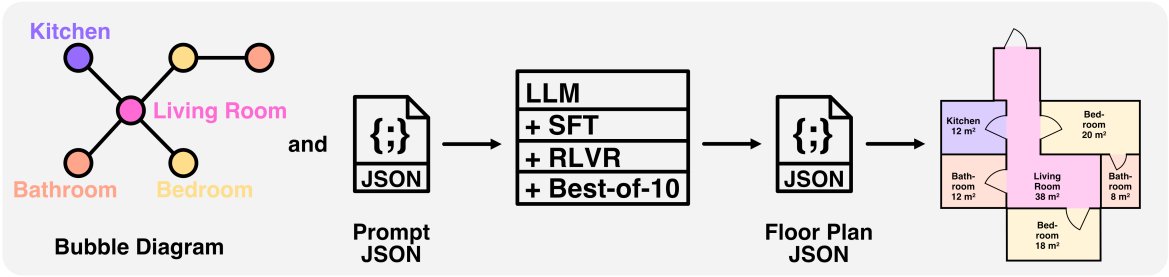

Figure 1: **Inference Process Overview.** Given a bubble diagram (input connectivity graph) and a JSON specification of design requirements (e.g., desired room sizes), our model generates a complete floor plan in JSON format.

Our key contributions are as follows:

1. We fine-tune an LLM in two stages, first via supervised learning and then via reinforcement learning, to transform constraint inputs into valid structured floor plans, demonstrating the feasibility and advantages of a structured-data-to-structured-data generative paradigm.

2. We propose new constraint adherence metrics that systematically evaluate the extent to which generated floor plans align with user-defined constraints, filling a critical evaluation gap in this domain.

2 Related Work

Generative Models. *House-GAN* and *House-GAN++* (Nauata et al., 2020, 2021) are a family of GAN-based methods that learn to generate floor plan images using convolutional graph-based networks. *House-GAN* starts with a noise vector for every existing room in the connectivity graph and makes use of a convolutional message passing network (Conv-MPN) to update node features while preserving spatial relationships. Notably, however, these models only condition on a bubble diagram and cannot impose other forms of constraints. The outputs from these methods are also scale-invariant (i.e., not metric) and thus cannot be used directly in any downstream task. *FloorplanGAN* (Luo and Huang, 2022) proposes a self-attention-based GAN that takes as input room centers, desired areas (encoded as relative proportions), as well as room type for each room. However, the outputs from the GAN do not always adhere to the original constraints. While the use of a differentiable rasterizer to compute losses in pixel space opens up possibilities, the core method does not appear to support a partial specification of constraints or polygons. *HouseDiffusion* (Shabani et al., 2023) is a diffusion-based method that creates a 1-D polygonal loop for each room in the bubble diagram and, through the diffusion process, iteratively improves their shape and position. While it also utilizes a transformer architecture to attend to the input graph, the floor plan generation process is done through diffusion. This method also only allows the input of a bubble diagram without the additional ability to condition on numerical constraints and therefore suffers from the same shortcomings as the *House-GAN* models.

Large Language Models. *ArchiText* (Galanos et al., 2023) also leverages LLMs to generate floor plans, but its prompts appear to be limited to natural-language descriptions rather than explicit geometry. In contrast to these approaches, our method allows users to specify detailed room constraints (e.g., room areas) and outputs room polygons in metric units that can directly be used in CAD drawing software. Providing both of these is crucial to the novelty and architectural usability of our method.

In *Tell2Design* (Leng et al., 2023), likely the most similar existing work, the authors create a new dataset for floor plan generation based on natural-language descriptions, based on *RPLAN*. They incorporate spatial (sizing) and relational positioning constraints in their natural-language prompts, and train a T5 sequence-to-sequence model on their dataset. Unlike our work, the authors render the floor plans as images and use image-based metrics for evaluation, which functionally ignore structural errors such as overlaps. Additionally, the evaluation of constraint adherence is somewhat limited, with human evaluation of only 100 test samples along 4 different axes (room type, room location, room size, and room relationships). In their evaluation, they show that their trained T5 model struggles particularly with spatial relationships. In this

work, we propose a robust set of complete metrics for constraint satisfaction that do not rely on rasterization, but rather measure the desired properties directly from the sequence output.

3D Scene Generation. Recently, full 3D scene generation methods have shown impressive results. *AnyHome* (Wen et al., 2023) and *Holodeck* (Yang et al., 2024) can generate floor plans, windows, doors, furniture, and meaningful placement in 3D, all from a natural-language prompt (e.g., "a 1b1b apartment of a researcher who has a cat"). In our method, we focus only on the floor plan aspect, but we allow for the specification of room dimensions and areas, as well as the total floor plan area, which neither method supports.

Constraint Satisfaction and Symbolic Methods. Earlier work formulates floor plan generation as a constraint satisfaction problem in which room geometries are represented by variables such as position and size, and layouts are generated by enforcing predefined spatial and dimensional rules (Medjdoub and Yannou, 2000; Li et al., 2000; Upasani et al., 2020). Related work by Lopes et al. (2010) proposes a procedural approach that utilizes relative area targets rather than explicit dimensional constraints, and organizes the layout through a hierarchical system of zones, such as public and private areas, which restricts direct topological relations to rooms within the same zone. These methods rely on hand-crafted rules and manually defined structures rather than learned priors from data. In contrast, our method learns plausible layouts directly from real floor plans while still conditioning on explicit structured constraints.

Datasets. One of the most commonly used datasets for floor plan generation is *RPLAN* (Wu et al., 2019). We use the dataset in this work, and we describe it in Section 3.1. A limitation of *RPLAN* for our setting is that it stores floor plans as images rather than in a vector format. As a result, it must first be converted into an intermediate structured representation.

3 Representation Format

Our task requires the model to satisfy explicit numerical and topological constraints and produce artifacts that can be automatically validated and consumed by downstream geometry and CAD tooling. Prior work shows that imposing structure in the *conditioning signal* can improve reliability: standardized marker templates provide an explicit control interface that improves generation quality and instruction adherence (D'souza et al., 2025). Complementarily, work in semantic parsing and code generation shows that unconstrained free-form generation is prone to syntactically invalid or non-executable outputs, while enforcing structural constraints during generation improves validity and accuracy (Yin and Neubig, 2018; Scholak et al., 2021; Raspanti et al., 2025). Collectively, these results substantiate the use of explicit, structured representations for both the conditioning and output sides.

Accordingly, we choose JSON over alternative input and output formats such as natural-language specifications. JSON (1) enables unambiguous parsing of numerical constraints, reducing ambiguity in measurements and spatial relationships, (2) enforces a consistent structure across training examples, reducing the burden on the model to infer implicit fields, (3) naturally captures nested relationships in floor plan specifications via its hierarchical organization, and (4) supports direct integration with CAD tools and architectural software that already rely on structured representations. See Appendix A.1 for the input and output schemas.

3.1 Dataset

RPLAN (Wu et al., 2019) contains 80,788 real floor plans from Asia stored as $256\times256\times4$ images that encode boundaries and room labels. We convert each plan to a JSON layout using the *HouseGAN++* data reader[2] and a custom converter that reconstructs room polygons, assigns semantic labels, scales coordinates to meters, computes geometric attributes, and derives a bubble-diagram adjacency list from interior-door connectivity. We filter out samples with disconnected graphs or invalid polygons. Full preprocessing details appear in Appendix A.2.

4 Our Method

To enable the model to generate valid floor plans from explicit design specifications, we employ a two-stage training procedure: supervised fine-tuning in stage one, and reinforcement learning–based fine-tuning in the second stage using GRPO with rewards derived from automated quantitative metrics, together with a hard feasibility condition that assigns zero reward to invalid or over-

[2] https://github.com/sepidsh/Housegan-data-reader

lapping outputs.

For all experiments, we use Llama-3.3-70B-Instruct as the backbone (Grattafiori et al., 2024). At inference time, we use **best-of-10 sampling**, selecting for each prompt the candidate with the smallest overlap area and breaking ties by Compatibility. Complete technical training and inference details appear in Appendix A.3.

4.1 Supervised Fine-Tuning

In the first phase, we perform supervised instruction tuning so that the model learns to translate structured prompts into JSON-encoded floor plans. Let $x = \{(k_i, v_i)\}_{i=1}^{K}$ be the conditioning input (e.g., room count, total area, bubble diagram), and let $y_{1:T}$ be the ground-truth token sequence. We adapt the pre-trained LLM by minimizing the negative log-likelihood:

$$L^{\mathrm{SFT}}(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[-\sum_{t=1}^{T}\log \pi_\theta(y_t \mid y_{<t}, x)\right]. \tag{1}$$

This objective drives the model to predict each token in y accurately given its prefix and the full specification x, producing floor plans that satisfy numerical and connectivity constraints.

After supervised fine-tuning, the LLM generates plausible floor plans but still produces overlapping polygons, a limitation that, to our knowledge, is not explicitly addressed by the methods we compare against and that must be addressed before deployment. We therefore introduce a second training stage to mitigate this issue.

4.2 GRPO

In the second phase of training, we perform reinforcement-learning-based fine-tuning using reinforcement learning with verifiable rewards (RLVR). Specifically, we use Group Relative Policy Optimization (GRPO) as our RLVR algorithm (Shao et al., 2024). GRPO is a PPO-style policy optimization method (Schulman et al., 2017) that improves the policy using relative reward comparisons within groups of sampled outputs, rather than relying on a learned value function.

For each input specification x, we sample a group of $G = 4$ candidate floor plans $\{y_i\}_{i=1}^{G}$ from the current policy. Each candidate is assigned a scalar reward $R(x, y_i)$ based on automatically computed constraint adherence metrics. We then normalize rewards within the sampled group to obtain group-relative advantages:

$$\hat{A}_i = \frac{R(x, y_i) - \mu_x}{\sigma_x + \epsilon}, \tag{2}$$

where μ_x and σ_x denote the mean and standard deviation of rewards over the sampled group for input x, and ϵ is a small constant for numerical stability.

The policy is then updated using a PPO-style surrogate objective:

$$L^{\mathrm{GRPO}}(\theta) = \mathbb{E}_{x\sim\mathcal{D}}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{\pi_\theta(y_i \mid x)}{\pi_{\theta_{\mathrm{old}}}(y_i \mid x)}\hat{A}_i\right]. \tag{3}$$

Intuitively, this objective increases the probability of candidates that perform better than the group average and decreases the probability of worse-performing candidates. By comparing outputs relative to one another for the same prompt, GRPO provides a stable training signal without requiring a separate critic network.

In our setting, the model generates multiple candidate floor plans for each input prompt. Any candidate that fails to parse as valid JSON or contains overlapping room polygons receives a reward of 0. Overlap is therefore handled as a hard feasibility condition rather than as a separate reward term. For each candidate y that parses as valid JSON and is free of overlapping room polygons, we compute two automatically verifiable reward terms:

1. **Connectivity Reward:** We reconstruct a connectivity graph from the generated layout and compare it with the input connectivity graph. The reward increases when the generated floor plan better matches the requested connectivity structure, and decreases when connectivity errors are present. This yields a reward $r_{\mathrm{conn}}(y) \in [0, 1]$, where higher is better, and 1 indicates an exact match.

2. **Total Area Reward:** Let $A(y)$ denote the total area of the generated layout for candidate y, and let $A^\star$ denote the target total area specified in the input prompt. We define the total area error as:

$$\mathrm{TAE}(y) = \frac{|A(y) - A^\star|}{A^\star}, \tag{4}$$

and the corresponding reward as:

$$r_{\mathrm{TA}}(y) = \max(0, 1 - \mathrm{TAE}(y)). \tag{5}$$

Model	Compatibility↓				Realism↑	Diversity↓			
Task	5	6	7	8	8	5	6	7	8
(Ashual and Wolf, 2019)	7.5	9.2	10.0	11.8	-1.00	120.6	172.5	162.1	183.0
(Johnson et al., 2018)	7.7	6.5	10.2	11.3	-1.00	167.2	168.4	186.0	186.0
House-GAN (Nauata et al., 2020)	2.5	2.4	3.2	5.3	-0.95	37.5	41.0	32.9	66.4
House-GAN++ (Nauata et al., 2021)	1.9	2.2	2.4	3.9	-0.52	30.4	37.6	27.3	32.9
HouseDiffusion (Shabani et al., 2023)	1.5	1.2	1.7	2.5	-0.19	11.2	10.3	10.4	9.5
(Ours)	**0.01**	**0.02**	**0.10**	**0.15**	**0.03**	**9.0**	**8.8**	**7.8**	**7.0**

Table 1: Main quantitative results comparing our approach with previous methods on Compatibility (↓), Realism (↑), and Diversity (↓). Our method fine-tunes a Llama-3.3-70B-Instruct in two stages: first supervised fine-tuning, followed by reinforcement learning with verifiable rewards. For our method, results are reported using best-of-10 sampling, selecting the candidate with the smallest overlap area and, in case of ties, the lowest Compatibility.

The final scalar reward is the equally weighted average of the Connectivity Reward and Total Area Reward. We found equal weighting to work well empirically, and leave a more systematic study of alternative reward weightings to future work.

5 Evaluation

We follow the evaluation protocol of previous work (Nauata et al., 2020, 2021; Shabani et al., 2023) and divide all floor plan samples into four groups based on the number of rooms (5, 6, 7, or 8 rooms). For each experiment, one group is held out completely, while the model is trained using 100% of the remaining three groups. The held-out group is split into two equal parts: 50% for validation and 50% for testing. This specific validation–test split is not explicitly defined in the original protocol. Still, we adopt it to provide a separate validation set for monitoring performance during training and a dedicated test set for final reporting.

For example, when performing the five-room task, all five-room samples are removed from the training set, and the model is trained on the six-, seven-, and eight-room samples. The five-room group is then split evenly for validation and testing. Following the evaluation protocol, all experiments are tested using a random subset of 1,000 samples from the test portion of the held-out group. This setup ensures that the model must generalize to unseen configurations rather than memorizing layouts for a specific room count.

We compare our method against bubble diagram constrained floor plan generators, including *House-GAN* (Nauata et al., 2020), *House-GAN++* (Nauata et al., 2021), and *HouseDiffusion* (Shabani et al., 2023), where *HouseDiffusion* represents the current peer-reviewed state of the art. We also compare against scene graph constrained image generation methods (Johnson et al., 2018; Ashual and Wolf, 2019).

5.1 Existing Metrics

We evaluate our method using three existing metrics: **Compatibility**, **Realism**, and **Diversity**, following prior work (Nauata et al., 2020, 2021; Shabani et al., 2023). **Compatibility** is computed as the graph edit distance (Sanfeliu and Fu, 1983) between the input bubble diagram and the bubble diagram reconstructed from the generated floor plan JSON. **Realism** is measured through feedback from a set of volunteers: each person views 10 randomized pairs of layouts (one ground truth and one generated) and indicates which layout appears more realistic, or whether both appear equally realistic. **Diversity** is measured using the Fréchet Inception Distance (FID) (Heusel et al., 2017), which compares the feature distributions of generated floor plan images and ground truth floor plan images. A more detailed description of these three metrics is provided in Appendix A.4.

5.2 Our Metrics

We also introduce four metrics that directly evaluate adherence to prompt-specified constraints and quantify polygon overlaps. To our knowledge, this is the first work to report metrics that explicitly measure both overlaps and room-size constraints. For all four metrics, we report the mean and standard deviation across samples.

- **Room Area ↓:** Mean absolute percentage error of per-room area with respect to the prompt.

Experiment	Task	Room Area↓	Room ID↑	Overlap↓	% Overlap↓	Compatibility↓	Diversity↓
Few-shot	5	0.27±0.22	0.96±0.19	0.55±0.50	0.07±0.10	2.93±1.26	45.96±0.00
SFT		**0.10±0.08**	1.00±0.00	0.12±0.33	0.00±0.02	0.02±0.16	**8.60±0.00**
SFT + RLVR		0.11±0.08	**1.00±0.00**	**0.03±0.17**	**0.00±0.00**	**0.01±0.14**	8.96±0.00
Few-shot	6	0.19±0.19	1.00±0.00	0.54±0.50	0.06±0.09	4.27±1.47	40.09±0.00
SFT		**0.10±0.07**	1.00±0.00	0.14±0.35	0.00±0.02	0.04±0.23	**7.59±0.00**
SFT + RLVR		0.12±0.08	**1.00±0.00**	**0.05±0.21**	**0.00±0.01**	**0.02±0.17**	8.79±0.00
Few-shot	7	0.15±0.14	0.99±0.05	0.50±0.50	0.05±0.08	5.27±1.19	41.73±0.00
SFT		**0.09±0.06**	1.00±0.00	0.23±0.42	0.01±0.02	0.17±0.51	**6.79±0.00**
SFT + RLVR		0.12±0.07	**1.00±0.00**	**0.09±0.29**	**0.00±0.01**	**0.10±0.40**	7.79±0.00
Few-shot	8	0.12±0.10	0.91±0.20	0.51±0.50	0.04±0.06	6.89±0.71	49.84±0.00
SFT		**0.08±0.05**	1.00±0.00	0.37±0.48	0.01±0.03	0.41±0.73	**6.44±0.00**
SFT + RLVR		0.10±0.06	**1.00±0.00**	**0.13±0.33**	**0.00±0.01**	**0.15±0.48**	6.96±0.00

Table 2: Task-wise results for Llama-3.3-70B-Instruct under few-shot with three examples, supervised fine-tuning, and supervised fine-tuning plus reinforcement learning with verifiable rewards. SFT + RLVR yields the lowest Overlap, % Overlap, and Compatibility across all tasks, while maintaining similar Room Area performance. All results in this table use best-of-10 sampling, with selection based on the smallest overlap area and, in case of ties, the lowest Compatibility.

- **Room ID ↑:** Exact-match accuracy of `id` relative to the prompt. Each `id` encodes both room type and instance index (e.g., "bedroom|1").
- **Overlap ↓:** Boolean indicator of whether any generated room polygons overlap.
- **% Overlap ↓:** Total overlapped area divided by the generated total area.

6 Experiments

6.1 Quantitative Evaluations

Table 1 reports the main quantitative results; for completeness, Appendix A.5 reproduces this table with standard deviations. Baseline numbers are taken from *HouseDiffusion* (Shabani et al., 2023). Our method achieves the lowest Compatibility score across all tasks (0.01 to 0.15), indicating near-perfect alignment with the input bubble diagram. For Diversity, it also attains the lowest values (7.0 to 9.0). On the eight-room task, relative to *HouseDiffusion*, our method reduces Compatibility by **94.00%** (from 2.50 to 0.15) and improves Diversity by **26.32%** (from 9.5 to 7.0).

Table 2 reports our metrics across training settings. After the second training stage (SFT + RLVR), which is our best-performing model, Room Area remains between 0.10 and 0.12, meaning the average per-room area deviates by only about 10% to 12% from the requested values and does not degrade as task complexity increases. Room ID is 1.00, indicating perfect labeling of both room type and instance-level identifiers (as specified in the prompt) across all tasks. However, Overlap increases as the room count increases (0.03, 0.05, 0.09, 0.13). The % Overlap is near zero across all tasks, indicating that, for a typical generated floor plan, the overlapped area is only a very small fraction of that plan's total area.

6.2 Qualitative Evaluations

Figure 2 compares our method with *HouseDiffusion* for the same input bubble diagram and shows a reference dataset sample. Across the five examples, our method matches the requested connectivity in every case, preserving the prompt-specified bubble diagram more reliably (i.e., showing a lower Compatibility score).

Beyond connectivity, our method produces floor plans whose room shapes and per-room proportions more closely match the reference samples. For instance, in the third row of Figure 2, the study, kitchen, and bathroom connected to the living room all have sizes similar to those in the reference.

We also verified this empirically by collecting feedback from a set of volunteers following the evaluation setup outlined in *HouseDiffusion*. The feedback setup included 10 pairwise comparisons and one warm-up example. Each comparison

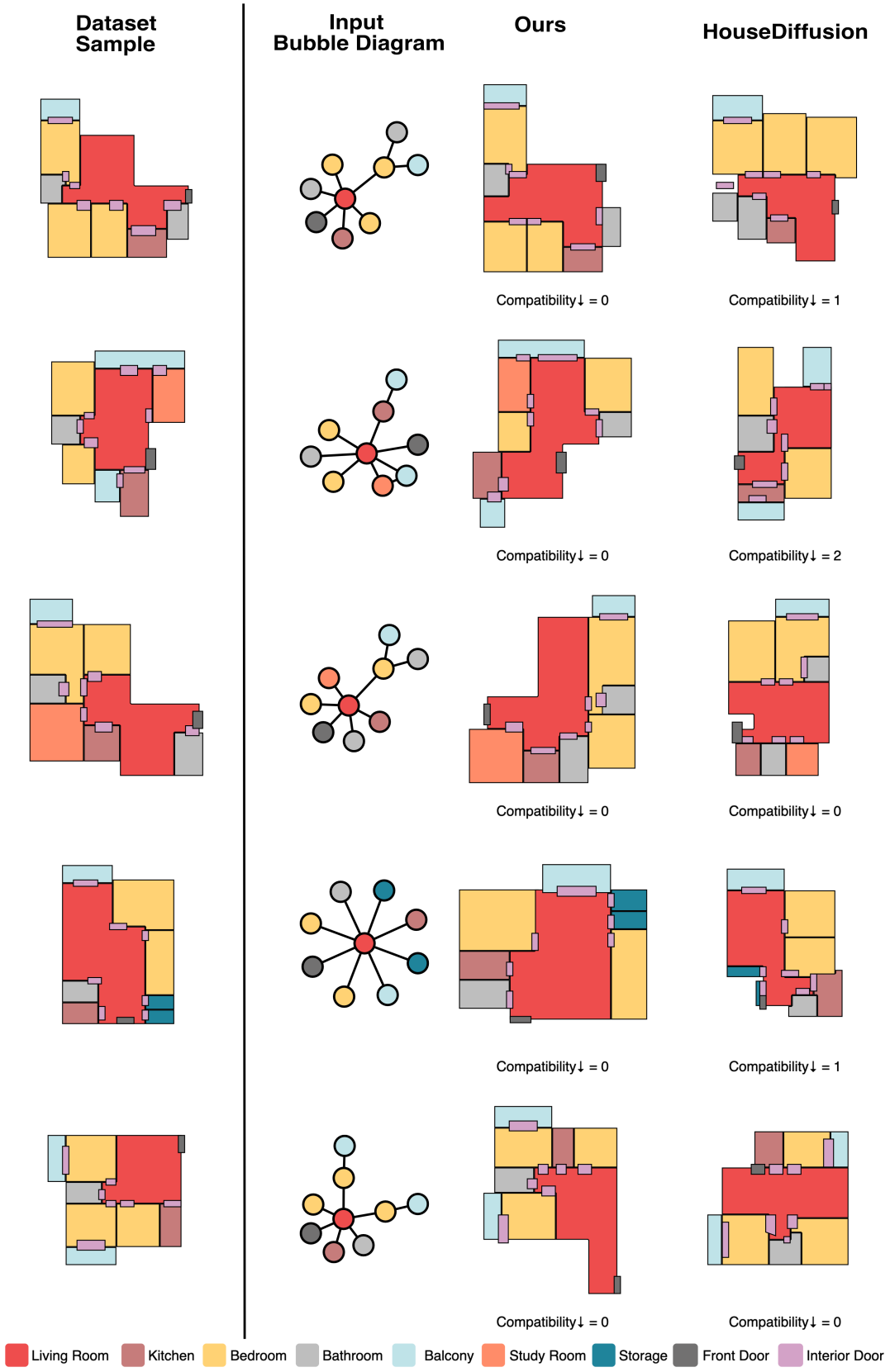

Figure 2: Qualitative comparison against *HouseDiffusion*. We replicate *HouseDiffusion*'s visualization to enable direct visual comparison. From left to right: dataset sample (reference), input bubble diagram, and layouts generated by our method and *HouseDiffusion* from the same bubble diagram. Numbers under the outputs report Compatibility (graph edit distance; lower is better). Our method better matches the reference room shapes and per-room proportions while achieving equal or lower Compatibility across the shown examples.

Experiment	Task	Budget	Room Area↓	Room ID↑	Overlap↓	% Overlap↓	Compatibility↓
SFT + RLVR	8	1	$\mathbf{0.09 \pm 0.05}$	1.00 ± 0.00	0.26 ± 0.44	0.01 ± 0.02	1.89 ± 1.97
		10	0.10 ± 0.06	1.00 ± 0.00	0.13 ± 0.33	0.00 ± 0.01	0.15 ± 0.48
		100	0.09 ± 0.06	$\mathbf{1.00 \pm 0.00}$	$\mathbf{0.10 \pm 0.30}$	$\mathbf{0.00 \pm 0.01}$	$\mathbf{0.02 \pm 0.16}$

Table 3: Effect of generation budget n on the eight-room task for Llama-3.3-70B-Instruct after two-stage fine-tuning. For each prompt, we sample $n \in \{1, 10, 100\}$ candidates and select the one with the least overlap, breaking ties by Compatibility.

showed a ground-truth floor plan and a floor plan generated by our method for the same bubble diagram. The order of comparisons and the position of the ground-truth and generated floor plans were randomized (e.g., sometimes the GT is on the left side, sometimes on the right). We asked a set of 40 volunteers who did not know about the project and had no prior experience designing floor plans to provide feedback. Volunteers were asked which layout appeared more realistic: (a) the ground truth (-1 point), (b) the generated one (+1 point), (c) both equally (0 points). The Realism score in Table 1 was calculated as the mean point score across all comparisons and volunteers. A screenshot of the feedback interface and the instructions given to volunteers is provided in Appendix A.6.

Based on feedback from 40 volunteers, our model achieves a **Realism score of 0.028**, a value very close to zero that indicates the generated layouts were judged similarly to the ground truth on average. This compares favorably with the Realism scores reported in prior work (see Table 1; this value is shown as 0.03 in the table after rounding to two decimal places).

6.3 Effect of the Training Stages

Table 2 compares three settings for Llama-3.3-70B-Instruct: few-shot prompting, supervised fine-tuning, and supervised fine-tuning followed by reinforcement learning with verifiable rewards. Few-shot prompting with three examples serves only as a baseline. It is unstable and scales poorly with task complexity: Compatibility rises from 2.93 in the five-room task to 6.89 in the eight-room task, while Overlap remains around 0.50 to 0.55 with high variance, even under best-of-10 selection. The high Diversity scores reflect uncontrolled variability, as many samples violate the bubble diagram, contain overlaps, or both.

After supervised fine-tuning, the model learns the mapping from structured prompts to JSON floor plan and already achieves low Room Area with perfect Room ID. However, as the room count increases, overlaps and connectivity mismatches also increase. Averaged across the five- to eight-room tasks, adding RLVR reduces Overlap by 65% and Compatibility by 56% relative to supervised fine-tuning. Meanwhile, % Overlap remains near zero, Room Area stays between 0.10 and 0.12, and Room ID remains perfect, indicating that these gains do not come at the expense of size accuracy or room labeling. Figure 3 highlights this progression across training stages.

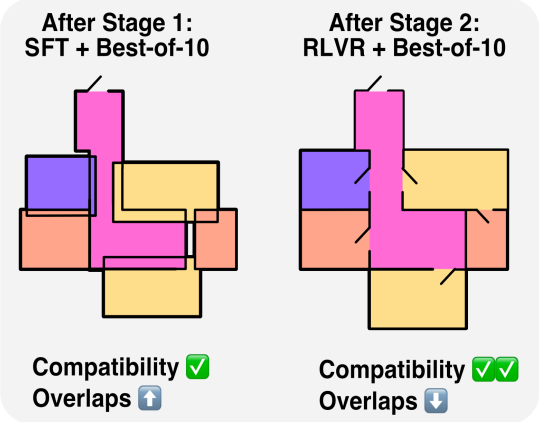

Figure 3: **Illustration of Post-Training Results.** Both Stage 1 (supervised fine-tuning, SFT) and Stage 2 (reinforcement learning with verifiable rewards, RLVR) use best-of-10 sampling. After Stage 1, the model generally follows the input bubble diagram but can still produce overlapping polygons. Stage 2 improves Compatibility and reduces Overlap.

6.4 Effect of the Generation Budget

We study the effect of the generation budget n on the eight-room task using best-of-n selection. For each prompt, we sample n candidates with temperature 0.7 and top-p 0.9, then choose the candidate with the smallest overlap area, using Compatibility to break ties. Table 3 reports results for $n \in \{1, 10, 100\}$. Increasing the budget from 10 to 100 reduces Overlap by 23.1% and Compatibility

Input Type	Room Area↓	Room ID↑	Overlap↓	% Overlap↓	Compatibility↓	Diversity↓
JSON	0.10 ± 0.06	$\mathbf{1.00 \pm 0.00}$	$\mathbf{0.13 \pm 0.33}$	$\mathbf{0.00 \pm 0.01}$	$\mathbf{0.15 \pm 0.48}$	$\mathbf{6.96 \pm 0.00}$
Natural-Language	0.10 ± 0.06	$\mathbf{1.00 \pm 0.00}$	0.14 ± 0.35	$\mathbf{0.00 \pm 0.01}$	0.37 ± 0.85	7.37 ± 0.00

Table 4: Comparison between structured JSON and natural-language inputs on the eight-room task for our best-performing model (**SFT** + **RLVR**) using best-of-10 sampling. We fine-tune the model only on JSON inputs, while the natural-language inputs come from deterministic verbalizations of the same constraints using three templates.

by 86.7%, while % Overlap remains at 0.00 ± 0.01. However, sampling 100 candidates per prompt substantially increases inference cost.

6.5 Zero-Shot Generalization to Natural-Language Constraints

Although our primary interface is structured JSON, we also evaluate whether the same model can handle natural-language interaction. To do so, we compare two input types: the original JSON constraints and natural-language descriptions obtained through deterministic verbalizations of the same information. To introduce limited phrasing variation without changing the meaning, we use three verbalization templates. The model is fine-tuned only on JSON inputs and is not retrained on natural-language prompts.

Table 4 shows results on the eight-room task for our best-performing model (SFT + RLVR) using best-of-10 sampling. Using natural-language inputs instead of JSON leads to almost no change in Room Area, Room ID, Overlap, or % Overlap. The main difference is in Compatibility, which rises from 0.15 ± 0.48 with JSON inputs to 0.37 ± 0.85 with natural-language inputs. One possible explanation is that connectivity constraints are more sensitive to wording, so verbalized descriptions introduce some ambiguity that is not present in the structured format. At the same time, the model's overall stability across the two input types suggests that it still retains the ability to interpret free-form constraint descriptions.

Overall, these results show that the model can handle natural-language inputs in a zero-shot setting for this task, while structured JSON remains the more reliable interface when precise connectivity control is required.

7 Conclusions

We introduced a structured generator that maps a bubble diagram and numerical specifications to a floor plan. Our two-stage training recipe, which combines supervised fine-tuning with reinforcement learning with verifiable rewards, trains an LLM to improve Compatibility while reducing overlap between room polygons and preserving area-related accuracy. Across the five- to eight-room tasks, our model achieves the lowest Compatibility (0.01 to 0.15), the lowest Diversity values (7.0 to 9.0), and a Realism score of 0.028, indicating that the generated layouts were judged similarly to the ground truth on average according to volunteer feedback. In the most complex eight-room setting, our method reduces Compatibility by 94% and improves Diversity by 26% relative to the peer-reviewed state of the art, *HouseDiffusion*. Together, these results show that text-based models can produce controllable, CAD-ready floor plans under explicit connectivity constraints and structured area specifications.

Limitations include occasional overlaps at higher room counts, inference requires multiple samples per prompt (increasing computational cost), and the use of single-floor residential floor plans from Asia derived from *RPLAN*, potentially limiting generalizability to other architectural styles or building types. Future work will scale to multi-floor residential buildings, add additional verifiable objectives to the reward (for example, circulation and daylight proxies), and enable interactive editing while preserving constraints. More broadly, the same structured approach should transfer to other graph and schema-constrained design problems beyond floor plan design.

Limitations

Our reinforcement learning with verifiable rewards (RLVR) optimizes only a limited set of automatically checkable objectives. These rewards are necessary but not sufficient for architectural validity and usability: many constraints that determine real-world quality (e.g., circulation, egress, accessibility,

daylight, structural constraints, and local building codes) are not modeled or verified. Consequently, strong performance on our metrics should not be interpreted as evidence of code compliance or construction readiness.

Our experiments are conducted on floor plans derived from RPLAN, which contains single-floor residential layouts from a specific geographic and cultural distribution. This scope limits generalization to other regions, architectural styles, multi-family housing, commercial programs, or multi-floor buildings. In addition, the conversion pipeline from rasterized annotations to metric polygons may introduce noise (e.g., imperfect polygon reconstruction or scaling). Our results implicitly assume these artifacts are not dominant; performance may degrade on datasets with different annotation conventions, room taxonomies, higher geometric complexity, or more complex constraint sets.

Finally, our empirical evidence is restricted to a single underlying dataset and a small set of task regimes (held-out room-count generalization). Although we evaluate on 1,000 test samples per task, we do not provide extensive ablations across multiple random seeds, alternative backbones, or alternative reward weightings, and we do not provide an extensive study of sensitivity to inference-time compute budgets beyond Section 6.4.

Acknowledgments

We thank the P2 program at Mila – Quebec AI Institute, Mila's IDT team for technical support, CIFAR for support through the Canada CIFAR AI Chair program, and NSERC for support under the Discovery Grants program.

References

Oron Ashual and Lior Wolf. 2019. Specifying object attributes and relations in interactive scene generation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Best Paper Honorable Mention.

Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Kiana Ehsani, Jordi Salvador, Winson Han, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. 2022. Procthor: Large-scale embodied ai using procedural generation. In *Advances in Neural Information Processing Systems*, volume 35.

Daniel D'souza, Julia Kreutzer, Adrien Morisot, Ahmet Üstün, and Sara Hooker. 2025. Treasure hunt: Real-time targeting of the long tail using training-time markers. *arXiv preprint*. ArXiv:2506.14702v1.

Theodoros Galanos, Antonios Liapis, and Georgios N Yannakakis. 2023. Architext: Language-driven generative architecture design. *arXiv preprint arXiv:2303.07519*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 540 others. 2024. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, volume 30, pages 6626–6637.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Justin Johnson, Agrim Gupta, and Li Fei-Fei. 2018. Image generation from scene graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1219–1228.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. *arXiv preprint arXiv:2309.06180*.

Sicong Leng, Yang Zhou, Mohammed Haroon Dupty, Wee Sun Lee, Sam Joyce, and Wei Lu. 2023. Tell2Design: A dataset for language-guided floor plan generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14680–14697, Toronto, Canada. Association for Computational Linguistics.

Siu-Pan Li, John H. Frazer, and Ming-Xi Tang. 2000. A constraint based generative system for floor layouts. In *Proceedings of the Fifth Conference on Computer Aided Architectural Design Research in Asia (CAADRIA 2000)*, pages 441–450, Singapore. Centre for Advanced Studies in Architecture (CASA).

Ricardo Lopes, Tim Tutenel, Ruben M. Smelik, Klaas Jan de Kraker, and Rafael Bidarra. 2010. A constrained growth method for procedural floor plan generation. In *Proceedings of GAME-ON 2010*, Leicester, United Kingdom.

Ziniu Luo and Weixin Huang. 2022. Floorplangan: Vector residential floorplan adversarial generation. *Automation in Construction*, 142:104470.

Benachir Medjdoub and Bernard Yannou. 2000. Separating topology and geometry in space planning. *Computer-Aided Design*, 32(1):39–61.

Nelson Nauata, Kai-Hung Chang, Chin-Yi Cheng, Greg Mori, and Yasutaka Furukawa. 2020. House-gan: Relational generative adversarial networks for graph-constrained house layout generation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 162–177. Springer.

Nelson Nauata, Sepidehsadat Hosseini, Kai-Hung Chang, Hang Chu, Chin-Yi Cheng, and Yasutaka Furukawa. 2021. House-gan++: Generative adversarial layout refinement network towards intelligent computational agent for professional architects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13632–13641.

Federico Raspanti, Tanir Ozcelebi, and Mike Holenderski. 2025. Grammar-constrained decoding makes large language models better logical parsers. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pages 485–499. Association for Computational Linguistics.

Alberto Sanfeliu and King-Sun Fu. 1983. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 13(3):353–362.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9895–9901. Association for Computational Linguistics.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *Preprint*, arXiv:1707.06347.

Amin Mohammad Shabani, Sepidehsadat Hosseini, and Yasutaka Furukawa. 2023. Housediffusion: Vector floorplan generation via a diffusion model with discrete and continuous denoising. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5466–5475.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Nitant Upasani, Krishnendra Shekhawat, and Garv Sachdeva. 2020. Automated generation of dimensioned rectangular floorplans. *Automation in Construction*, 113:103149.

Zehao Wen, Zichen Liu, Srinath Sridhar, and Rao Fu. 2023. Anyhome: Open-vocabulary generation of structured and textured 3d homes. *arXiv preprint arXiv:2312.06644*.

Wenming Wu, Xiao-Ming Fu, Rui Tang, Yuhan Wang, Yu-Hao Qi, and Ligang Liu. 2019. Data-driven interior plan generation for residential buildings. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 38(6).

Yue Yang, Fan-Yun Sun, Luca Weihs, Eli VanderBilt, Alvaro Herrasti, Winson Han, Jiajun Wu, Nick Haber, Ranjay Krishna, Lingjie Liu, Chris Callison-Burch, Mark Yatskar, Aniruddha Kembhavi, and Christopher Clark. 2024. Holodeck: Language guided generation of 3d embodied ai environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16227–16237.

Pengcheng Yin and Graham Neubig. 2018. TRANX: A transition-based neural abstract syntax parser for semantic parsing and code generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (System Demonstrations)*, pages 7–12, Brussels, Belgium. Association for Computational Linguistics.

A Appendix

A.1 JSON Data Schemas for Floor Plan Generation

In this section, we describe the exact JSON schemas our model consumes and produces. Each schema is built around a top-level `spaces` array whose elements represent any floor plan entity, including both rooms and doors (interior or front).

In the input schema, each space must include an `id` and a `room_type` label, and may specify either an explicit `area` for irregular-polygon shapes or a pair of `height` and `width` values for rectangular spaces. The input also includes a `room_count`, a `total_area` constraint, and an `input_graph` dictionary encoding the bubble diagram; front doors must be specified explicitly in the `spaces` array, while interior door connections are inferred from the bubble diagram. The input schema is summarized in Table 5.

In the output schema, each space object includes its computed `area` and a `floor_polygon` array of vertices defining its precise footprint in absolute coordinates. All area values (`area`, `total_area`) are given in square meters. The output schema is summarized in Table 6.

A.2 RPLAN Conversion

RPLAN (Wu et al., 2019) is a manually collected dataset of 80,788 real-world floor plans of buildings in Asia. Each floor plan in *RPLAN* is stored as a $256 \times 256 \times 4$ four-channel image. Channels 1 and 2 store interior and exterior boundary information; channel 3 contains room information where

Field	Description
`room_count`	Total number of rooms
`total_area`	Sum of all room areas
`spaces`	Array of space objects
`id`	Unique identifier (e.g., `"bedroom|0"`)
`room_type`	Semantic label (e.g., `"bedroom"`)
`area`	Area for an irregular polygon space (omit `height` and `width`)
`height`	Height of bounding rectangle for a regular polygon space (omit `area`)
`width`	Width of bounding rectangle for a regular polygon space (omit `area`)
`input_graph`	Bubble diagram. Each key is a space ID mapping to an array of its neighbor IDs

Table 5: Input JSON data structure for floor plan generation.

Field	Description
`room_count`	Total number of rooms in the generated floor plan
`total_area`	Sum of all generated room areas
`spaces`	Array of space objects
`id`	Unique identifier (e.g., `"bedroom|0"`)
`room_type`	Semantic label (e.g., `"bedroom"`)
`area`	Area of the polygon space
`floor_polygon`	List of vertices outlining the space polygon
`x`	X–coordinate
`y`	Y–coordinate

Table 6: Output JSON data structure for generated floor plans.

each pixel value denotes which room it belongs to; channel 4 has extra information to distinguish rooms with the same room type value in channel 3.

To convert this 4-channel image into a JSON structure, we first use the same data-reader as in *House-GAN++*[3], then process each entry with a custom converter that maps room codes to semantic names, reconstructs room polygons from boundary segments, and computes geometric attributes such as area, width, and height. The polygon vertices are expressed in absolute coordinates after scaling from pixels to meters. Each floor plan is converted into a set of spaces, each assigned a unique identifier and associated numeric attributes (e.g., area, width, height), together with the bubble diagram encoded as an adjacency list. We obtain each bubble diagram from the interior door connectivity graph produced by our *RPLAN* conversion pipeline and represent it as an adjacency list stored in a JSON dictionary. Each key represents a room identifier, and each value is an array of room identifiers that are directly connected via interior doors. The front door is modeled as a special space that connects to exactly one room. Any sample whose adjacency list is disconnected or that contains rooms lacking valid polygons is filtered out.

Applying this pipeline yields four room-count specific datasets: 8-room with 53,001 training, 8,596 test, and 8,597 validation examples; 7-room with 44,859 training, 12,667 test, and 12,668 validation; 6-room with 47,955 training, 11,119 test, and 11,120 validation; and 5-room with 65,000 training, 2,597 test, and 2,597 validation.

The **RPLAN** dataset is released under a restricted-access, research-only data-use agreement. It is constructed from anonymized floor plans that remove user and privacy information, and the authors state that the underlying plans were curated to avoid copyright issues. Access is granted only via the authors' official request process, and the terms explicitly limit use to non-commercial research and academic purposes and prohibit redistribution of the dataset in any way or format (in whole or in part). Accordingly, we do not redistribute RPLAN, nor any processed versions, subsets, annotations, or other derivative data products that contain any portion of the original dataset. Any work derived from RPLAN must follow the same conditions, including non-commercial academic use only and no redistribution; new users must obtain access directly through the authors' official channel.

A.3 Training and Inference Details

In the first training stage, we fine-tune the Llama-3.3-70B-Instruct (Grattafiori et al., 2024) backbone using 4-bit quantization and adapter-based PEFT (LoRA) (Hu et al., 2021). We configure LoRA with rank $r = 64$, $\alpha = 128$, dropout 0.1, and a learning rate of $1e - 4$. Training is distributed across a 6-node Slurm cluster (each node: 4 × NVIDIA H100 80 GB). We pack the examples into a context window of 6k tokens, use a device batch size of 2, and train for two epochs. For the most complex 8-room floor plan generation task, this requires up to four hours, and the same amount of time or less on smaller-room tasks.

We initialize the second stage from the supervised fine-tuning checkpoint, obtained by merging the SFT LoRA adapter into the base model. We then train with GRPO using TRL[4]. Training runs

[3] https://github.com/sepidsh/Housegan-data-reader

[4] https://github.com/huggingface/trl

Model	Compatibility↓			
Task	5	6	7	8
(Ashual and Wolf, 2019)	7.5±0.0	9.2±0.0	10.0±0.0	11.8±0.0
(Johnson et al., 2018)	7.7±0.0	6.5±0.0	10.2±0.0	11.3±0.1
House-GAN (Nauata et al., 2020)	2.5±0.1	2.4±0.1	3.2±0.0	5.3±0.0
House-GAN++ (Nauata et al., 2021)	1.9±0.3	2.2±0.3	2.4±0.3	3.9±0.5
HouseDiffusion (Shabani et al., 2023)	1.5±0.0	1.2±0.0	1.7±0.0	2.5±0.0
(Ours)	**0.01±0.1**	**0.02±0.2**	**0.10±0.4**	**0.15±0.5**

Table 7: Compatibility (↓) results (mean ± std) across tasks.

Model	Realism↑	Diversity↓			
Task	8	5	6	7	8
(Ashual and Wolf, 2019)	-1.00	120.6±0.5	172.5±0.2	162.1±0.4	183.0±0.4
(Johnson et al., 2018)	-1.00	167.2±0.3	168.4±0.4	186.0±0.4	186.0±0.4
House-GAN (Nauata et al., 2020)	-0.95	37.5±1.1	41.0±0.6	32.9±1.2	66.4±1.7
House-GAN++ (Nauata et al., 2021)	-0.52	30.4±4.4	37.6±3.3	27.3±4.9	32.9±4.9
HouseDiffusion (Shabani et al., 2023)	-0.19	11.2±0.2	10.3±0.2	10.4±0.4	9.5±0.1
(Ours)	**0.03**	**9.0±0.0**	**8.8±0.0**	**7.8±0.0**	**7.0±0.0**

Table 8: Realism (↑) and Diversity (↓) results (mean ± std) across tasks.

on a 7-node Slurm cluster with 4 NVIDIA H100 80 GB GPUs per node: 6 nodes are used for distributed optimization, and 1 node hosts a dedicated vLLM server for rollout generation (Kwon et al., 2023). We use a per-device batch size of 1 and sample 4 generations per prompt. Optimization uses AdamW with learning rate 10^{-6}, clipping parameter 0.2, and KL coefficient 0.04. Rollouts are generated with temperature 0.9, top-p 1.0, and maximum prompt and completion lengths of 4096 tokens. Validation is performed every 100 steps on 200 sampled validation examples, and checkpoints are selected based on validation reward. In practice, the best model is typically the first checkpoint evaluated at step 100, which on our hardware corresponds to at most about 2 hours of wall-clock time per task. This makes the GRPO stage shorter than supervised fine-tuning.

At inference time, unless otherwise stated, we use temperature 0.7, top-p 0.9, and a best-of-10 strategy. Candidates are ranked by minimum overlap area and, in case of ties, minimum Compatibility score.

Below is the exact prompt we use to condition the model:

```
<|begin_of_text|><|start_header_id|>system<|end_header_id|>You
      are a state-of-the-art floor-plan generator that
      translates JSON specifications and connectivity
      requirements defined by a bubble diagram into precise,
      optimized layouts.
Your algorithm considers each room's dimensions, proportion,
      and desired adjacencies to produce an efficient
      arrangement that maximizes usable space while honoring
      all constraints.
Your top priority is that no two room polygons ever overlap.
      Rooms must be strictly disjoint, doors may touch room
      boundaries, but room interiors must never intersect.
Your output must be a JSON object, where `output` key contains
      :
- `room_count`: the total number of room entries
- `spaces`: a list of mixing rooms and doors. Each room or
      door entry must include:
   - `id`: formatted as `<room_type>|<unique_index>` (e.g. `"
      bedroom|2"` or `"interior_door|0"`)
   - `room_type`: the room type (e.g. `"living_room"`, `"
      kitchen"`, etc.)
   - `area` in square meters (all positive numbers)
   - `floor_polygon`: an ordered list of `{x: , y:}` vertices
      defining a simple polygon
Additional rules:
- **Absolute non-overlap**: no two room polygons may share any
       interior point under any circumstances.
- Every adjacency in the bubble diagram must be bridged by
      exactly one door.
- Every `id` used in the bubble diagram and on any door must
      appear in the `rooms` list.
Return only a JSON object containing an `output` key without
      extra commentary or explanation.<|eot_id|>
```

A.4 Compatibility, Realism and Diversity Metrics

Compatibility is computed as the graph edit distance (Sanfeliu and Fu, 1983) between the input bubble diagram and the bubble diagram reconstructed from the generated floor plan JSON. A

lower score indicates higher consistency with the specified connectivity, with a score of 0 meaning a perfect match. In this sense, Compatibility can be interpreted as the number of connectivity mistakes in the generated floor plan, making it the most direct measure of whether the model satisfies the user-defined connectivity constraints.

Realism is measured through feedback from a set of volunteers: each person reviews 10 randomized pairs of layouts (one ground truth, one generated) and selects the more realistic layout, or indicates that both are equally realistic. For each pair, we assign $+1$ if the volunteer selects the generated floor plan, -1 if they select the ground-truth floor plan, and 0 if they judge them equally realistic. We report the mean of these scores across all comparisons and all volunteers. Values near zero indicate that, on average, generated layouts are indistinguishable from ground truth.

Diversity is measured using the Fréchet Inception Distance (FID) (Heusel et al., 2017), which compares the feature distributions of generated floor plan images and ground truth floor plan images. Rather than relying on pixel-level differences, FID computes the mean and covariance of deep feature representations to assess the similarity between the two distributions. A lower FID means that the generated floor plans have feature statistics closer to those of the ground truth, indicating that the model captures both the diversity and the overall distribution of the reference data more effectively. We compute FID using a custom visualization pipeline that mimics the *HouseDiffusion* visualizer on our data.

A.5 Full Main Results with Standard Deviations

This section reproduces Table 1 and adds standard deviations. Table 7 reports mean $\pm$ standard deviation for Compatibility on the five to eight-room tasks, using the same evaluation setup as in the main results. Table 8 reports mean $\pm$ standard deviation for Realism and Diversity under the same setup; the only difference from the main results is the inclusion of dispersion values for completeness.

A.6 Realism Feedback Setup

Instructions provided to volunteers at the beginning of the Realism feedback exercise:

> *Please help us decide if the AI-generated floor plans look as realistic as the ground-truth ones.*
>
> *What we want you to do is for each of the following 10 floor plans to decide if the one on the left or the one on the right looks more realistic, or they both look more or less the same.*
>
> *Notes:*
>
> - *There are doors between rooms (marked in purple) and entrance doors (marked in dark gray) that are placed pretty arbitrarily, even in the GT dataset.*
> - *There are gaps between rooms (also known as "walls") of varying thickness, even within the same floor plan. This is normal.*
> - *Floor plans only very rarely end up perfectly square, so don't look for that.*
> - *Important: We are not asking you to detect which ones are AI-generated. We are asking you to pick which one you find more plausible or realistic.*
> - *Important: Both being equal is a perfectly fine response.*
> - *We will not elaborate what "realistic" or "plausible" means. Just use your best judgment.*

Figure 4 shows one example comparison from the feedback interface. We recruited postgraduate student volunteers through our academic network and did not provide monetary compensation. The feedback exercise was anonymous and did not collect personally identifiable information.

A.7 Potential Risks and Mitigations

Automating parts of the floor plan design process may contribute to job displacement or devaluation of professional expertise in architecture, drafting, real estate, and related services. As such systems become more capable, organizations may be incentivized to reduce human involvement, shifting labor demand away from early-stage drafting and toward fewer, higher-leverage roles. **Mitigation:** We position the system as an assistive tool intended to augment professionals rather than replace them. In any deployment, we recommend (i) treating outputs as drafts that require expert review and sign-off, (ii) providing training and upskilling resources to help

practitioners integrate generative tools into established workflows, and (iii) designing interfaces that preserve human control (e.g., editable constraints, transparent validation reports, and explicit handoff points to licensed professionals).

Generated floor plans may fail to satisfy building codes, accessibility requirements, structural constraints, or domain-specific best practices, even when they satisfy the limited constraints we verify. If used without expert oversight, such outputs could contribute to unsafe designs, construction errors, and downstream legal liability, and could negatively affect occupants' quality of life. **Mitigation:** We recommend human-in-the-loop use with clear disclaimers that outputs are not certified designs. For higher-stakes settings, the pipeline should incorporate rigorous automated checks (e.g., egress and circulation, accessibility, minimum clearances, code rule sets), post-generation geometric repair, and professional review by qualified architects and engineers.

A.8 Behind the scenes

In this section, we guide readers through the complete research process, including the experiments and approaches that did not work, sharing insights and lessons that may benefit other researchers.

ProcTHOR. We explored augmenting our training dataset with synthetic houses from ProcTHOR (Deitke et al., 2022), whose floor plan generator first samples an outer shell by iterative boundary cuts and then subdivides it into rooms using the recursive layout method of (Lopes et al., 2010). In that procedural generator, spaces are organized hierarchically into functional zones (public vs. private) before being split into individual rooms, which aligns with the typical day/night zoning principle.

Despite these attractive properties, qualitative inspection revealed frequent topological pathologies that conflict with our target distribution: for example, bathrooms acting as corridors connecting a kitchen and a bedroom, or four-bedroom layouts where one must pass through multiple bedrooms to reach the last. Given the prevalence of such issues, we elected not to use ProcTHOR as a data source and relied instead on curated real-world layouts.

Llama-3.1-8B-Instruct. We first tried Llama-3.1-8B-Instruct as a backbone. Even with very large LoRA adapters (rank $r \in \{256, 512\}$ and $\alpha \in \{128, 256\}$), the model did not yield reliable results. At inference time, it often fell into a repetition loop that emitted the same value sequence for one polygon vertex array within a room, producing degenerate or invalid geometry. Because this failure mode persisted, we discarded the 8B variant for this application and adopted a Llama-3.3-70B-Instruct backbone instead.

Few-shot Prompting Alone Is Not Enough. We evaluated few-shot prompting with state-of-the-art backbones, including GPT-4o, OpenAI o3, and QwQ-32B. None consistently produced a valid floor plan JSON. Few-shot prompting alone does not provide the built-in structure needed to enforce bubble diagram connectivity and numerical constraints jointly. Even with careful prompt engineering and schema exemplars, outputs remained brittle. Typical failures included non-closed polygons, self-intersections, duplicated or missing rooms, violations of bubble diagram connectivity, numerical drift in room areas, repetition loops that copied a single coordinate across a polygon, and schema hallucinations.

We therefore adopted supervised fine-tuning, followed by reinforcement learning with verifiable rewards. That said, we do not rule out the possibility that future models with stronger structured generation capabilities may make few-shot prompting alone sufficient for this task.

Reward Hacking. We initially optimized only the Connectivity Reward. The model quickly learned to game this objective: it collapsed geometry into tiny rooms so that the reconstructed adjacency matched the prompt, yielding near-zero Compatibility while violating requested areas and hurting Realism.

Figure 4: Example comparison from our Realism feedback exercise, showing a ground truth floor plan and a generated one from our method for the same bubble diagram. In this case, ground truth is left. The order was randomized across comparisons.