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

Luis Lara
Mila
luis.lara@mila.quebec

Zhi Hao Luo
Mila

Aristides Milios
Mila & Polytechnique Montréal

Aditya Sharma
Mila & Polytechnique Montréal

Ge Ya Luo
Mila & Université de Montréal

Christopher Beckham
Mila

Florian Golemo
Mila

Christopher Pal
Mila & Polytechnique Montréal
Canada CIFAR AI Chair

Abstract

An AI system for professional floor plan design needs to be able to precisely control room dimensions and areas (quantitative constraints), while also balancing functional considerations and design aesthetics. Existing generative approaches focus primarily on respecting the requested connectivity between rooms, but do not support generating floor plans with 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 enforce both numerical (areas, dimensions) and spatial (topological) constraints. Furthermore, we design a set of constraint adherence metrics to measure how generated floor plans align with user-defined constraints systematically. Our model generates floor plans that satisfy numerical constraints and outperforms existing methods on realism, compatibility, and diversity scores. Specifically, our approach leads to an up to 94% reduction in compatibility score. Our results demonstrate that LLMs can effectively handle quantitative constraints in structured design tasks, suggesting broader applications for text-based generative modeling.

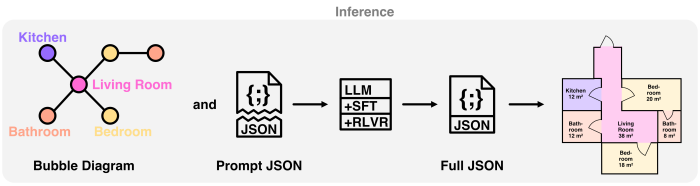
1 Introduction

North American housing systems face persistent affordability pressures. In the United States, credible estimates of the national housing shortfall range from 3.7 million to 4.9 million homes, depending on methodology [Freddie Mac Economic, 2024, Patel et al., 2024]. In Canada, the national housing agency projects that restoring affordability by 2030 would require roughly 3.5 million additional homes beyond current trajectories, and 2022 data indicate that about one in three renter households spent at least 30% of income on shelter [Canada Mortgage and Housing Corporation, 2023, Statistics Canada, 2024].

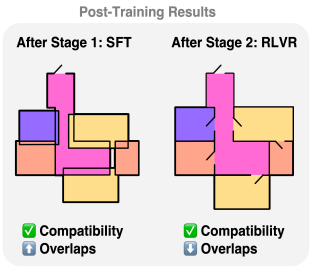
Multiple factors drive the gap between need and delivery. High and volatile construction and financing costs have pushed many projects below feasibility, making them unlikely to be built and weakening future supply and affordability [Garcia, 2023]. Early-stage floor plan decisions are part of this landscape. They are not the definitive bottleneck, but they can influence unit yield, constructability, and program compliance, and they often require rapid iteration with public agencies, funders, and community stakeholders.

Generative approaches for layouts have advanced quickly, but many methods output raster images that are difficult to edit, validate, or connect to downstream rule checking. Vectorized approaches

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: UrbanAI.



(a) Inference Process Overview.



(b) Illustration of Post-Training Results.

Figure 1: **Overview. Left:** Given a bubble diagram (input connectivity graph) and an architectural constraints JSON (for example, desired room sizes), our model outputs a JSON floor plan with metric-scale polygons that are directly renderable. **Right:** After Supervised Fine-Tuning (SFT), the plans respect the bubble diagram yet exhibit polygon overlaps; adding Reinforcement Learning with Verifiable Rewards (RLVR) reduces these overlaps while preserving connectivity.

address some of these issues by modeling rooms and boundaries as structured objects. However, prior graph-conditioned methods such as *HouseGAN* and *House-GAN++* enforce constraints only through an input adjacency graph (a bubble diagram). They produce axis-aligned room layouts but do not impose quantitative constraints such as room areas [Nauata et al., 2020, 2021]. Likewise, *HouseDiffusion* is conditioned only on the bubble-diagram adjacency graph and generates vector layouts by diffusing 2D room and door coordinates [Shabani et al., 2023]. Datasets like *RPLAN* have enabled learning at scale [Wu et al., 2019].

We target this gap with a JSON-based representation that encodes room layouts as polygonal structures with explicit attributes. The representation supports fine-grained control of spatial parameters, such as room sizes and connectivity. We fine-tune a large language model (LLM) to generate JSON-encoded plans from prompts that specify spatial constraints. See Figure 1a for a schematic of the pipeline. We evaluate constraint satisfaction using proposed spec-adherence metrics and observe high compliance across tasks. In the most complex setting, eight-room generation with best-of-10 sampling, our approach yields up to a 94% reduction in the compatibility score (Table 1).

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, and (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.

We use JSON rather than free-form natural language because it provides unambiguous parsing of numeric and spatial constraints, enforces a consistent schema across training examples, mirrors the hierarchical structure of floor plan specifications, and could integrate with CAD and architectural software. See Appendix A.1 for the schemas and Appendix A.2 for the dataset preparation details.

2 Method

Stage 1: Supervised fine-tuning. We begin with supervised instruction, tuning the model to translate structured prompts into JSON floor plans. The conditioning input is a set of key-value pairs (for example, room counts, target total area, and a bubble diagram), and the target is the ground-truth token sequence. We adapt a pre-trained LLM by minimizing token-level negative log-likelihood over the dataset. This stage yields plausible floor plans that satisfy most numerical and connectivity requirements, yet overlapping polygons persist, a limitation unaddressed by prior work that we mitigate in the next stage.

Stage 2: RLVR with GRPO. Our second training stage performs reinforcement-learning-based fine-tuning with verifiable rewards (RLVR), using GRPO [Shao et al., 2024]. GRPO is a variant of PPO Schulman et al. [2017] that groups multiple trajectories together and optimizes relative

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, and uses a best-of-10 sampling strategy.

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**	**-0.01**	**9.0**	**8.8**	**7.8**	**7.0**

improvements within the group. By using group-relative statistics, GRPO does not require the use of a critic in the same way PPO does (which belongs to the actor-critic family of RL methods).

$$\hat{A}(x, y) = \frac{R(x, y) - \mu_x}{\sigma_x} \tag{1}$$

$$L^{\text{GRPO}}(\theta) = \mathbb{E}_{x \sim \mathcal{D}} \left[\sum_{y \in \mathcal{Y}(x)} \frac{\pi_\theta(y|x)}{\pi_{\theta_{\text{old}}}(y|x)} \cdot \hat{A}(x, y) \right] \tag{2}$$

Equation 1 defines the group-relative advantage function, which normalizes each candidate's reward by comparing it to other outputs for the same input, where μ_x and σ_x are the mean and standard deviation of rewards within group $\mathcal{Y}(x)$. Equation 2 is the simplified GRPO objective. For each input x, GRPO samples a group of candidate outputs $\mathcal{Y}(x)$ from the old policy and computes z-score normalized advantages within that group. The policy is then optimized using the importance sampling ratio from PPO, weighted by these group-relative advantages. Candidates with above-average rewards receive positive advantages and larger updates, while below-average candidates receive negative advantages. This simplified form omits: (1) PPO clipping on the importance ratio and (2) KL regularization to a reference policy. Complete technical training and inference details appear in Appendix A.3.

Rewards for floor-plan generation. In our case, we generate multiple floor plans from the model, and assign them automatically calculated rewards. We use two rewards with equal weight: (1) compatibility score, graph edit distance between the input connectivity graph and the generated plan's connectivity graph, and (2) $1 - \text{TAE}$ (the total area error), which increases as the generated total area approaches the target area in the prompt.

3 Our metrics

To evaluate generated floor plans, we introduce four metrics that directly test 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:

- **Room Area ↓:** Mean absolute percentage error of per-room `area` with respect to the prompt.
- **Room ID ↑:** Exact-match accuracy of `room_id` relative to the prompt. Each `room_id` encodes both room type and instance index (e.g., `"bedroom|1"`).
- **Overlap ↓:** Boolean check that any two generated room polygons overlap.
- **Percentage Overlap ↓:** Total overlapped area as a percent of the generated `total_area`.

We also evaluate our method using three additional metrics: **Compatibility**, **Realism**, **Diversity**, following prior work [Nauata et al., 2020, 2021, Shabani et al., 2023]. **Compatibility** is computed as

the graph edit distance [Abu-Aisheh et al., 2015] between the input bubble diagram and the bubble diagram reconstructed from the generated JSON. **Realism** is measured in a human study. **Diversity** is measured using the Fréchet Inception Distance [Heusel et al., 2017]. A complete description of these three metrics is provided in Appendix A.4.

4 Evaluation

We follow prior work [Nauata et al., 2020, 2021, Shabani et al., 2023] and group samples by room count (5, 6, 7, 8). For each task, one group is held out, the model is trained on the remaining three, and the held-out group is split evenly into evaluation and test. Full details are in Appendix A.5. 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], and scene graph constrained image generation with Johnson et al. [2018] and Ashual and Wolf [2019]. *HouseDiffusion* Shabani et al. [2023] is the current peer-reviewed state of the art.

5 Experiments

5.1 Quantitative evaluations

Table 1 reports the main quantitative results; for completeness, Appendix A.6 reproduces this table with standard deviations. Baseline numbers are copied 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 best values (7 to 9). Relative to *HouseDiffusion*, our method improves compatibility by **94%** and diversity by **26.32%** on the eight-room generation task, the most complex setting. Our metrics across both training stages are reported in Appendix Table 2, with a detailed discussion in Section A.7. See Figure 1b for a visual illustration of the stage-wise effects.

5.2 Qualitative evaluations

Appendix Figure 2 compares our method with *HouseDiffusion* for the same input bubble diagram. 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 yields room shapes and per-room proportions that more closely match the reference floor plans. For example, in the third row of Appendix Figure 2, the study, kitchen, and bathroom connected to the living room have sizes similar to the reference. We validated this with a single-blind user study following *HouseDiffusion*. Thirteen graduate students unfamiliar with the project and without floor-plan design experience completed 10 trials plus a warm-up. Each trial showed a ground-truth plan and one generated by our method for the same bubble diagram; question order and left/right placement were randomized. Participants selected which looked more realistic: ground truth (-1), generated ($+1$), or both equally (0). The realism score was the mean across all trials and participants. We obtained -0.008 across $n = 13$, a value close to zero that indicates our plans are nearly indistinguishable from ground truth and compare favorably to prior reports (Table 1; rounded to fit the table format). Survey details are in Appendix A.9.

6 Conclusions

We introduced a structured-to-structured generator that maps a bubble diagram and numeric specifications to CAD-ready floor plans via a two-stage recipe that combines supervised instruction tuning with reinforcement learning with verifiable rewards. The model satisfies quantitative and topological constraints while minimizing polygon overlaps, and achieves state-of-the-art results across room-generation tasks: compatibility of 0.01 to 0.15, diversity of 7 to 9, and a realism score of -0.01, with a 94% reduction in compatibility error and a 26% diversity gain in the eight-room setting relative to *HouseDiffusion*. Remaining limitations include occasional overlaps at higher room counts and a focus on single-floor residential Asian plans derived from *RPLAN*. Future work will extend to multi-floor layouts, incorporate additional verifiable objectives such as circulation and daylight proxies, and enable constraint-preserving interactive editing.

References

Zeina Abu-Aisheh, Romain Raveaux, Jean-Yves Ramel, and Patrick Martineau. An exact graph edit distance algorithm for solving pattern recognition problems. In *4th International Conference on Pattern Recognition Applications and Methods 2015*, 2015.

Oron Ashual and Lior Wolf. Specifying object attributes and relations in interactive scene generation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019. URL https://arxiv.org/abs/1909.05379. Best Paper Honorable Mention.

Canada Mortgage and Housing Corporation. Housing shortages in canada: Updating how much housing we need by 2030, 2023. URL https://assets.cmhc-schl.gc.ca/sites/cmhc/professional/housing-markets-data-and-research/housing-research/research-reports/2023/housing-shortages-canada-updating-how-much-we-need-by-2030-en.pdf. Baseline estimate of 3.5 million additional homes needed by 2030.

Freddie Mac Economic. Economic, housing and mortgage market outlook — november 2024: Spotlight on housing supply, November 2024. URL https://www.freddiemac.com/research/forecast/20241126-us-economy-remains-resilient-with-strong-q3-growth. Estimates U.S. housing shortage at 3.7 million units as of Q3 2024.

David Garcia. Making it pencil: The math behind housing development — 2023 update, December 2023. URL https://ternercenter.berkeley.edu/research-and-policy/making-it-pencil-2023/.

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

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

Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1219–1228, 2018. URL https://arxiv.org/abs/1804.01622.

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

Nelson Nauata, Kai-Hung Chang, Chin-Yi Cheng, Greg Mori, and Yasutaka Furukawa. 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, 2020.

Nelson Nauata, Sepidehsadat Hosseini, Kai-Hung Chang, Hang Chu, Chin-Yi Cheng, and Yasutaka Furukawa. 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, 2021.

Elena Patel, Aastha Rajan, and Natalie Tomeh. Make it count: Measuring our housing supply shortage, November 2024. URL https://www.brookings.edu/articles/make-it-count-measuring-our-housing-supply-shortage/. Updates U.S. shortage estimate to 4.9 million units for 2023.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

Amin Mohammad Shabani, Sepidehsadat Hosseini, and Yasutaka Furukawa. 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, 2023.

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

Statistics Canada. The daily — housing affordability in canada, 2022, September 2024. URL https://www150.statcan.gc.ca/n1/daily-quotidien/240910/dq240910b-eng.htm.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

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

A Technical appendices and supplementary material

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 (Table 3), 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. In the output schema (Table 4), 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.

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 $256 \times 256 \times 4$ vector image. Channels 1 and 2 store interior and exterior boundary information; channel 3 contains room information where 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 *HouseGAN++*[1], 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 (eg. 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.

A.3 Training and inference details

In the first stage of the training, we fine-tune the Llama-3.3-70B-Instruct [Touvron et al., 2023] backbone using 4-bit quantization and adapter-based PEFT (LoRA) [Hu et al., 2021]. We configure LoRA with rank $r = 64$ and $\alpha = 128$, 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 equal or less on smaller-room tasks.

In the second stage, we continue from the supervised fine-tuning checkpoint and train with reinforcement learning using GRPO with TRL[2]. Training is launched on a 7-node Slurm cluster (each node: 4 × NVIDIA H100 80 GB), 6 for model training, and 1 for inference. A dedicated vLLM server process handles fast generation [Kwon et al., 2023]. We use a per-device batch size of 1 and sample 4 generations per prompt. The best performance is obtained by saving the first checkpoint after 100 training samples; on our hardware, this corresponds to at most 2 hours of wall-clock time per task.

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

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

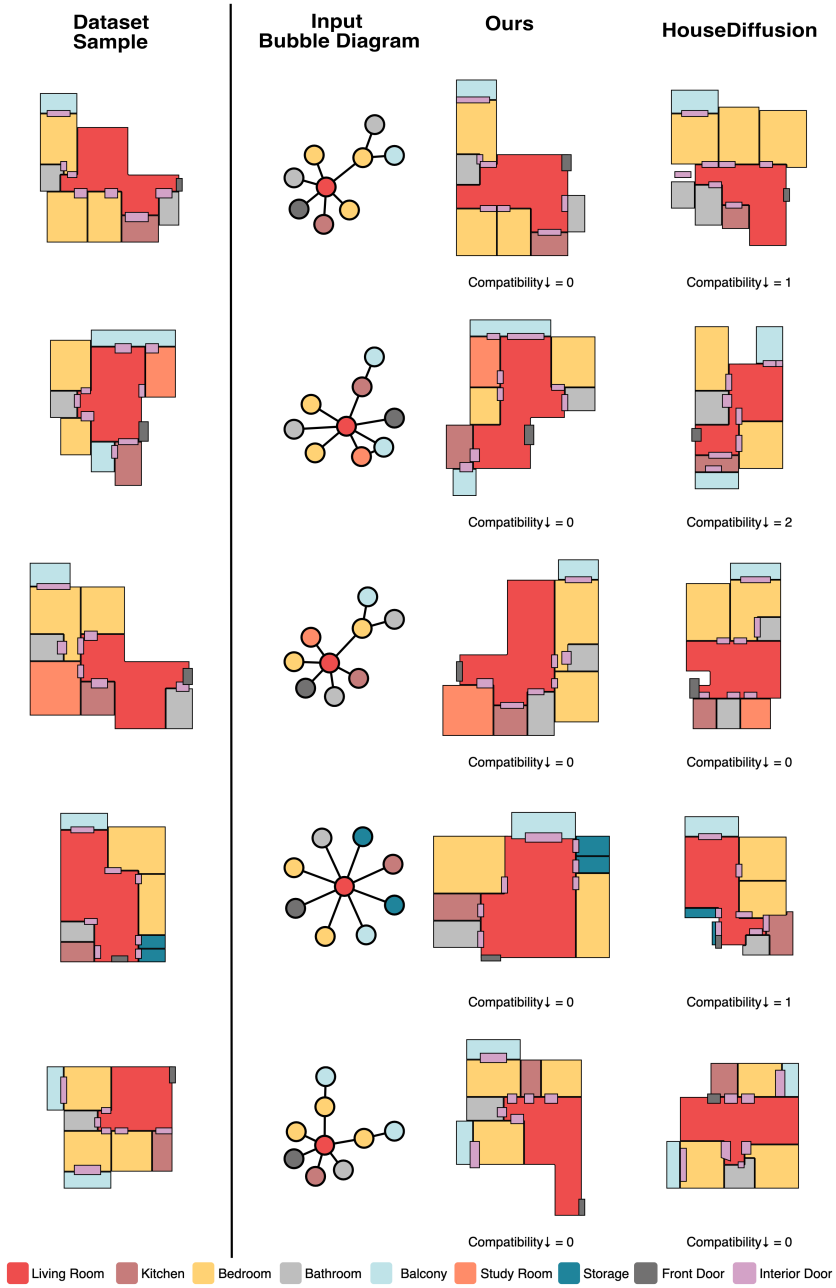


Figure 2: Qualitative comparison against *HouseDiffusion*. We replicate *HouseDiffusion*'s visualization to enable direct visual comparison. From left to right: dataset sample (reference), input bubble diagram, and layouts generated by our method and *HouseDiffusion* from the same bubble diagram. Numbers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

Thus, the second phase sees only 100 training examples in every task, which is far fewer than in supervised fine-tuning.

At inference time, we sample with a temperature of 0.7 and top-p of 0.9 under a best-of-n strategy to generate ten candidates, and select the candidate with the smallest overlap area and, in case of ties, the lowest 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

Compatibility is computed as the graph edit distance [Abu-Aisheh et al., 2015] between the input bubble diagram and the bubble diagram reconstructed from the generated 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 a human study in which each participant sees 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 score $+1$ if they select the generated floor plan, -1 if they select the ground-truth, and 0 if they judge them equally realistic. Summing these scores over all pairs and all participants yields a total realism score. 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 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 Evaluation protocol

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 evaluation and 50% for testing. This specific evaluation–test split is not explicitly defined in the original protocol, but we adopt it to provide a separate evaluation set for monitoring performance during training and a dedicated test set for final reporting.

For example, when performing the 5-room task, all 5-room samples are removed from the training set, and the model is trained on the 6-, 7-, and 8-room samples. The 5-room group is then split evenly for evaluation 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.

A.6 Full main results with standard deviations

This section reproduces the Table 1 and adds standard deviations. Table 5 reports mean ± standard deviation for compatibility, realism, and diversity on the five to eight-room tasks, using the same evaluation setup as the main results. The only change is the inclusion of dispersion values for completeness.

A.7 Effect of the training stages

Table 2 compares three settings for Llama 3.3 70B: few-shot prompting, supervised fine-tuning, and supervised fine-tuning followed by reinforcement learning with verifiable rewards. Few-shot with three examples serves as a backbone-only baseline. It is unstable and scales poorly with task size: compatibility rises from 2.93 at five rooms to 6.89 at eight rooms, while overlap stays saturated around 0.50–0.55 with huge variance, indicating frequent collisions even under best-of-10 selection. The very high diversity scores reflect uncontrolled variability, since many samples violate the bubble diagram or overlap constraints.

Stage one learns the mapping from structured prompts to JSON floor plan and already yields low room area error and perfect room identification, but as the room count grows, overlaps and bubble diagram mismatches increase. Averaged over tasks with five to eight rooms, adding RLVR cuts overlaps by 65% and reduces compatibility by 56% compared with supervised fine-tuning. Besides, percentage overlap is near zero and room area MAPE remains near 0.10 to 0.12, and room ID accuracy stays at 1, so these gains do not trade off size accuracy or room labeling.

Finally, we evaluate the effect of the number of generations at inference using a best of n selector on the eight-room task. For each prompt, we sample n candidates with temperature 0.7 and top p 0.9, then select the candidate with the smallest overlap area, breaking ties by lower compatibility. Appendix Table 6 reports results for $n \in \{1, 10, 100\}$. Moving from 10 to 100 samples improves overlap by 30% and compatibility by 89% while Percentage Overlap remains at 0.00 ± 0.01. However, generating 100 samples per prompt is computationally expensive and time-consuming in production environments, so the main paper reports the best of 10.

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.

Llama-3.1-8B-70B-Instruct. We first tried an 8B Llama-3.1-Instruct 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 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 70B Llama-3.3-Instruct backbone instead.

Prompting Alone Is Not Enough. We ran few-shot prompts with state-of-the-art backbones, including GPT-4o, OpenAI o3, and QwQ-32B. None consistently produced a valid floor plan JSON.

In short, few-shot prompting does not provide the built-in structure needed to enforce bubble diagram connectivity and numeric constraints jointly. Even with careful prompt engineering and schema exemplars, outputs remained brittle. Typical failures include non-closed polygons, self-intersections, duplicated or missing rooms, violations of bubble diagram connectivity and exterior door semantics, numerical drift in room areas, repetition loops that copy a single coordinate across a polygon, and schema hallucinations. We therefore adopted supervised fine-tuning, followed by reinforcement learning with verifiable rewards.

Reward Hacking. We initially optimized only the compatibility 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.

To remove this failure mode, we added a second verifiable term that rewards accurate total area, and trained with GRPO using equal weights on compatibility and total-area rewards (Section 4.2). After this change, room sizes stabilized (room-area MAPE 0.10–0.12 across tasks), and compatibility improved without encouraging degenerate geometry.

A.9 Realism survey design

Instructions given to participants at the start of the realism survey:

> *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 3 is a screenshot from one of the questions.

Figure 3: Example question from our user study, 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 randomly shuffled between questions.

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 is stage one supervised fine-tuning that teaches the model to map structured specifications to JSON floor plans. RLVR is stage two reinforcement learning with verifiable rewards, trained with GRPO to reduce polygon overlaps and compatibility errors. SFT + RLVR yields the lowest overlap, percent overlap, and compatibility while keeping room area MAPE and diversity low. Results use best-of-10 sampling, selecting the candidate with the smallest overlap area and, in case of ties, the lowest compatibility score.

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 3: Input JSON data structure for floor plan generation.

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 4: Output JSON data structure for generated floor plans.

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 5: 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 fine-tuning with GRPO, and uses a best-of-10 sampling strategy.

Model	Compatibility↓				Realism↑	Diversity↓			
Task	5	6	7	8	8	5	6	7	8
(Ashual and Wolf [2019])	7.5±0.0	9.2±0.0	10.0±0.0	11.8±0.0	-1.00	120.6±0.5	172.5±0.2	162.1±0.4	183.0±0.4
(Johnson et al. [2018])	7.7±0.0	6.5±0.0	10.2±0.0	11.3±0.1	-1.00	167.2±0.3	168.4±0.4	186.0±0.4	186.0±0.4
(Nauata et al. [2020])	2.5±0.1	2.4±0.1	3.2±0.0	5.3±0.0	-0.95	37.5±1.1	41.0±0.6	32.9±1.2	66.4±1.7
(Nauata et al. [2021])	1.9±0.3	2.2±0.3	2.4±0.3	3.9±0.5	-0.52	30.4±4.4	37.6±3.3	27.3±4.9	32.9±4.9
(Shabani et al. [2023])	1.5±0.0	1.2±0.0	1.7±0.0	2.5±0.0	-0.19	11.2±0.2	10.3±0.2	10.4±0.4	9.5±0.1
(Ours)	**0.01±0.1**	**0.02±0.2**	**0.10±0.4**	**0.15±0.5**	**-0.01**	**9.0±0.0**	**8.8±0.0**	**7.8±0.0**	**7.0±0.0**

Table 6: 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.

Experiment	Task	n	Room Area↓	Room ID↑	Overlap↓	% Overlap↓	Compatibility↓
Llama + 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	$\mathbf{1.00 \pm 0.00}$	$\mathbf{0.10 \pm 0.30}$	$\mathbf{0.00 \pm 0.01}$	$\mathbf{0.02 \pm 0.16}$