STARPLAN: STABLE TOPOLOGY-ALIGNED REPRESENTATIONS FOR AUTOREGRESSIVE VECTOR FLOORPLAN GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper presents a novel approach for vectorized floorplan generation from bubble diagrams using a topology-representation-based autoregressive framework. The framework learns topology representations from the complete input graph before generation and preserves them as stable structural conditions throughout Transformer decoding. A graph-derived generation order establishes correspondence between graph entities and generated spatial objects, allowing the learned representations to remain associated with their intended objects as the sequence unfolds. We find that a simple strategy of reusing and injecting the same topology representations as both complete graph conditions and aligned object identities effectively bridges unordered graph structures and standard autoregressive sequence modeling. This provides persistent access to complete topology while retaining entity-level correspondence during sequential generation. Evaluation on the RPLAN dataset demonstrates that our method achieves significant improvements over the state of the art across all evaluated settings and metrics.

1 INTRODUCTION

Automatic floorplan generation has been studied under building boundaries, room specifications, and bubble diagrams (Bao et al., 2013; Bisht et al., 2022; Wu et al., 2019; Hu et al., 2020). Bubble diagrams compactly specify room types and connectivity, making them useful conditions for layout generation (Nauata et al., 2020; 2021; Tang et al., 2023). Large-scale resources such as Zillow Indoor Dataset, FloorPlanCAD, and CubiCasa5K have also supported data-driven floorplan analysis and modeling (Cruz et al., 2021; Fan et al., 2021; Kalervo et al., 2019).

House-GAN and House-GAN++ synthesize rasterized room masks from room relations, while Graph2Plan combines a layout graph with a building boundary and geometric optimization (Nauata et al., 2020; 2021; Hu et al., 2020). Vector methods instead generate explicit room, door, or wall geometry (Liu et al., 2022; Para et al., 2021; Shabani et al., 2023; Sun et al., 2022). In particular, Liu et al. (2022) generate room and door coordinates autoregressively from a bubble diagram.

This task presents a mismatch: the input is an unordered graph, but the output is an ordered geometric sequence. Complete topology is known before decoding, whereas geometric context grows with the generated prefix. A decoder must use the former without losing the identity of the graph node associated with each new object.

We learn node representations from the complete bubble graph through masked graph pretraining before autoregressive geometry generation. These representations remain fixed as the floorplan sequence unfolds.

Functional Order (FO) converts graph nodes into a deterministic object sequence. The decoder accesses all node representations through cross-attention, while Graph-Aligned Representation (GAR) adds the corresponding node representation to each object's coordinate tokens. Thus complete topology and object identity are available throughout generation.

The main contributions of this work are summarized as follows:

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

- We learn graph node representations from complete input topology before geometry decoding and reuse them as stable conditions for autoregressive vector generation.
- FO and GAR preserve node-to-object correspondence, allowing the same representations to provide complete-graph context and object-specific identity.
- Experiments on RPLAN show improvements in realism, diversity, and topology compatibility across mixed and separate training settings.

2 Related Work

2.1 Graph-constrained floorplan generation

Graph2Plan combines a layout graph and building boundary to predict rasterized rooms and bounding boxes, then optimizes room alignment (Hu et al., 2020). House-GAN propagates information across connected and unconnected rooms; House-GAN++ adds iterative refinement (Nauata et al., 2020; 2021). GTGAN models local and global node relations with graph convolution and attention (Tang et al., 2023), while factor-graph approaches model higher-order constraints (Dupty et al., 2024).

Vectorized methods generate editable geometric primitives directly. Floor-SP reconstructs editable geometry from raster floorplan images through sequential room-wise shortest paths (Chen et al., 2019). WallPlan generates wall graphs (Sun et al., 2022); constraint-graph modeling generates vector layout elements autoregressively with subsequent optimization (Para et al., 2021). Liu et al. (2022) predict room and door bounding boxes with a graph-conditioned Transformer and refine the draft layout. HouseDiffusion generates polygonal room and door loops by graph-conditioned denoising (Shabani et al., 2023). More recent methods generate structural graphs (Hu et al., 2025), use graph and boundary conditions in diffusion (Hong et al., 2024), generate vectors from boundaries (Wang & Pajarola, 2025), or serialize several floorplan conditions in a shared markup language (Shiohara & Yamasaki, 2026).

Interactive and partial-input methods such as iPLAN and MaskPLAN address related design settings (He et al., 2022; Zhang et al., 2024). Our focus is complete bubble-graph conditioning for vector generation, with graph-node identities preserved during decoding.

2.2 Autoregressive models for layout generation

Autoregressive layout models serialize semantic and geometric attributes. LayoutVAE conditions scene layouts on categories (Jyothi et al., 2019); LayoutTransformer predicts primitive attributes with masked self-attention (Gupta et al., 2021); VTN adds variational modeling (Arroyo et al., 2021). PolyGen extends sequential prediction to quantized 3D mesh geometry (Nash et al., 2020).

For conditional generation, Para et al. (2021) use constraint graphs, and LayoutFormer++ serializes user constraints (Jiang et al., 2023). BLT relaxes fixed autoregressive order through bidirectional masked prediction (Kong et al., 2022). In the draft stage of Liu et al. (2022), graph propagation follows the generated prefix, so complete generated-layout context enters through a later refinement stage.

Our method instead conditions each autoregressive step on representations of the complete input graph and aligns them with generated objects through a graph-derived order.

3 Method

Our method converts a bubble diagram into a room-and-door graph, learns node representations from the complete graph, and generates vector bounding boxes autoregressively. The graph-derived object order aligns each predicted box with its node representation. Fig. 1 shows how complete-graph conditioning and node-specific alignment enter the decoder.

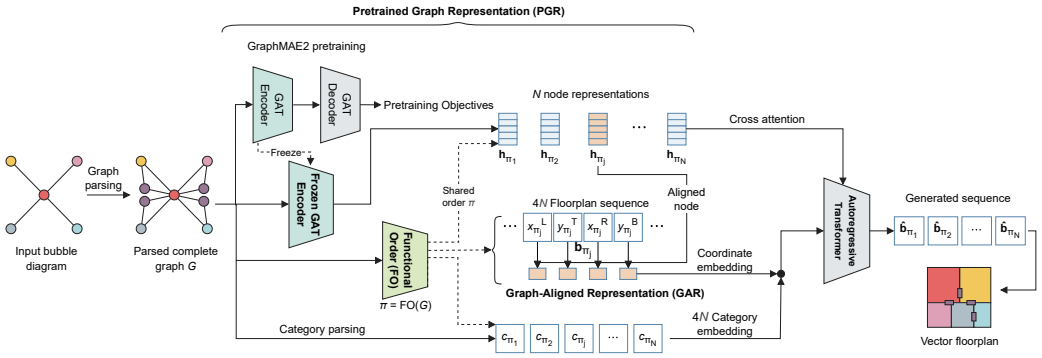


Figure 1: Overview of our method. The complete bubble graph is encoded by a frozen GraphMAE2 encoder into Pretrained Graph Representation (PGR). Functional Order (FO) establishes the graph-node/object order. During autoregressive decoding, the complete set of PGR conditions coordinate states, while Graph-Aligned Representation (GAR) adds the aligned node representation to the four coordinate tokens of each object.

3.1 Topology Representation Learning

Graph construction. Given an input bubble diagram, we construct an undirected attributed graph $G = (V, E, X)$, where each node represents a room, an interior door, or a front door. Each node feature $\mathbf{x}_i \in \{0, 1\}^{15}$ encodes its semantic category. Each room relation is converted into a door node connected to its incident rooms, and a self-loop is added to every node. The topology encoder uses only node semantics and connectivity, without bounding-box coordinates, raster positions, or generation order.

Masked graph autoencoding. We adopt GraphMAE2, building on masked graph autoencoding (Hou et al., 2022; 2023), to learn topology representations through self-supervised pretraining. A subset of node features is replaced by learnable mask tokens, while edges are randomly dropped to construct corrupted graph views. A two-layer graph attention network (GAT) encoder (Veličković et al., 2018) maps the masked graph to node representations:

$$\mathbf{H} = E_\theta(G_s, \widetilde{\mathbf{X}}) \in \mathbb{R}^{|V| \times 128}. \tag{1}$$

A GAT decoder reconstructs the semantic features of the masked nodes. Stochastic latent remasking and an EMA teacher are further employed to improve representation consistency. The encoder is pretrained using the masked feature reconstruction and latent consistency objectives of GraphMAE2 (Hou et al., 2023). After pretraining, we retain and freeze the encoder. The resulting node representations encode the complete input topology independently of geometry generation. We refer to this graph condition as **Pretrained Graph Representation (PGR)** and use it for the autoregressive decoder in Sec. 3.2.

3.2 Autoregressive Floorplan Generation with a Graph-Aligned Transformer

Layout representation. Given a bubble graph $G = (V, E)$, each node $v_i \in V$ corresponds to a room, an interior door, or a front door. Its geometry is represented by an axis-aligned bounding box:

$$\mathbf{b}_i = (x_i^L, y_i^T, x_i^R, y_i^B), \tag{2}$$

where (x_i^L, y_i^T) and (x_i^R, y_i^B) denote the top-left and bottom-right corners, respectively. All coordinates are normalized to a 256×256 canvas and quantized into 256 discrete bins. Given an object order π, the floorplan is represented as

$$\mathbf{s} = [\langle BOS \rangle, \mathbf{b}_{\pi_1}, \ldots, \mathbf{b}_{\pi_N}, \langle EOS \rangle]. \tag{3}$$

Each bounding box contains four consecutive coordinate tokens. The bubble diagram assigns each graph node v_i a semantic category c_i; after ordering, the category embedding of c_{π_j} is added to the four token positions of $\mathbf{b}_{\pi_j}$ as a condition. The graph-derived order π is introduced below.

Autoregressive decoding. We employ a causal Transformer (Vaswani et al., 2017) following the LLaMA architecture (Touvron et al., 2023) to generate the coordinate sequence from left to right. The conditional distribution is factorized as

$$p(\mathbf{s} \mid G) = \prod_{t=1}^{T} p_\theta(s_t \mid s_{<t}, G). \tag{4}$$

Causal self-attention models geometric dependencies among previously generated coordinates, while PGR provides node-level topology information from the complete input graph. At each step, the decoder predicts a categorical distribution over the 256 coordinate bins. Training uses teacher forcing, while inference starts from $\langle BOS \rangle$ and proceeds autoregressively until $\langle EOS \rangle$ is generated.

Graph conditioning. The pretrained graph representations describe the complete bubble graph before geometry decoding. At each decoder layer, we use cross-attention with the coordinate states as queries and the graph representations as keys and values:

$$\mathbf{C}^{(l)} = \mathrm{CrossAttn}(\mathbf{Z}^{(l)}, \mathbf{H}, \mathbf{H}). \tag{5}$$

This allows each coordinate state to retrieve relevant information from the complete graph throughout autoregressive generation.

Graph-aligned representation. We derive a Functional Order (FO) directly from the input graph:

$$\pi = FO(G) = (\pi_1, \ldots, \pi_N). \tag{6}$$

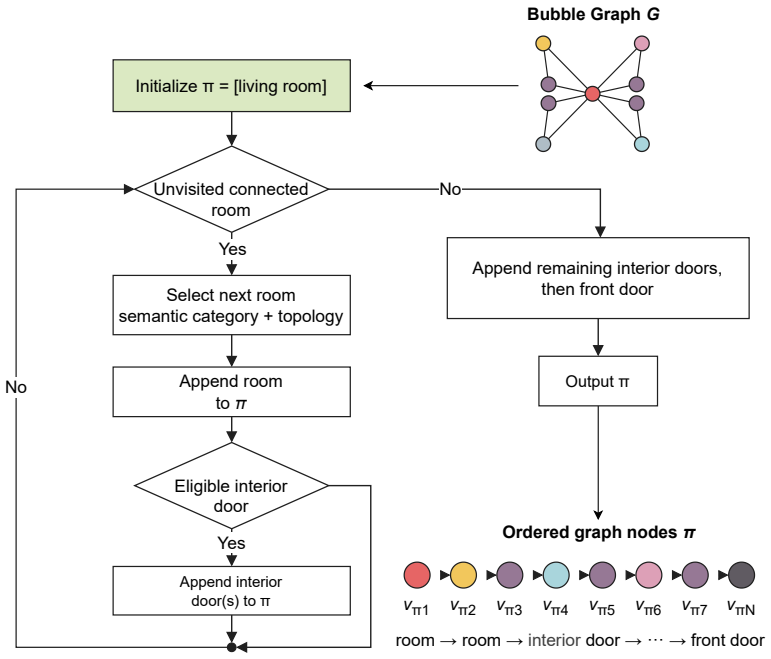


Figure 2: Functional Order (FO) turns the input graph into an object sequence. It starts at the living room, traverses connected rooms, inserts eligible interior doors, and places the remaining doors and front door last.

As Figure 2 illustrates, FO starts from the living room and visits connected rooms according to their semantic categories and topology. Interior doors are inserted after both incident rooms have been visited, while front doors are placed at the end. It thus provides a node-to-sequence correspondence available during both training and inference, aligning the j-th bounding box with graph node v_{π_j}.

Based on this correspondence, **Graph-Aligned Representation (GAR)** augments the coordinate tokens of each object with its aligned graph representation. Let $\mathcal{T}_j$ denote the four token positions of the j-th bounding box. Before entering the decoder, the input representation is augmented as

$$\widetilde{\mathbf{z}}_t = \mathbf{z}_t + \alpha M \mathbf{h}_{\pi_j}, \qquad t \in \mathcal{T}_j, \tag{7}$$

where M projects the graph representation to the decoder dimension and α is a learnable scalar. The four coordinates of the same object therefore share the representation of the corresponding graph node. PGR provides complete-graph information to the decoder, while GAR preserves node-specific correspondence during object generation.

3.3 Loss Functions

Reconstruction loss. We apply the standard cross-entropy loss to learn a categorical distribution over the quantized coordinates. The loss is averaged over the sequence length T:

$$\mathcal{L}_{rec} = -\frac{1}{T}\sum_{t=1}^{T} \log p_\theta(s_t \mid s_{<t}, G). \tag{8}$$

Geometric loss. The reconstruction loss supervises individual coordinate tokens but does not explicitly measure geometric relations between layout elements. Following prior vectorized floorplan generation work, we convert each predicted coordinate distribution into a continuous value by expectation, allowing the geometric objective to remain differentiable. We then construct the pairwise IoU matrices $\mathcal{G}_{gen}$ and $\mathcal{G}_{gt}$ for the generated and ground-truth layouts:

$$\mathcal{L}_{geo} = \frac{1}{N_l}\sum_{i=1}^{N_l} \left\| \mathcal{G}^i_{gen} - \mathcal{G}^i_{gt} \right\|_1, \tag{9}$$

where N_l denotes the number of element pairs. The final objective combines the two losses with equal weights:

$$\mathcal{L} = \mathcal{L}_{rec} + \mathcal{L}_{geo}. \tag{10}$$

4 Experiments

4.1 Dataset and preprocessing

Similar to prior vectorized floorplan generation work, we conduct experiments on the RPLAN dataset (Wu et al., 2019), which contains 80,788 residential floorplans. We extract room and door instances and convert their polygonal layouts into bounding boxes. This conversion may produce invalid door-room relations. We identify front doors overlapping room boxes using a threshold of $\tau = 50\%$ and correct their positions according to the adjacent room and exterior boundary. For each interior door, we examine the room instances on its opposite sides and remove samples where the door does not connect two distinct rooms. Layouts with disconnected bubble graphs or rooms completely occluded after bbox conversion are also excluded. After preprocessing, more than 60,000 valid layouts remain for the experiments. Representative cases are provided in Figure S1 of the supplementary material.

4.2 Baselines and evaluation metrics

We compare our method with representative approaches from three generative paradigms: LayoutTransformer serves as the autoregressive baseline, HouseGAN++ represents graph-constrained raster generation, and HouseDiffusion provides a graph-conditioned vector diffusion baseline. LayoutTransformer is adapted to receive the object category sequence without graph representation learning. The rasterized outputs of HouseGAN++ are converted into bounding boxes for evaluation. All baselines are retrained using their official implementations with the same pre-processing and data splits.

We evaluate realism, diversity, and compatibility. Following the ranking-based evaluation of Liu et al. (2022), adapted to our four methods, 10 architecture experts and 7 amateurs assessed 100 inputs. For each bubble diagram, participants ranked the four generated floorplans separately for visual quality and functional rationality. First through fourth place received $+2$, $+1$, -1, and -2, respectively; Table 1 reports the mean scores. The supplementary material details the protocol. We measure diversity by Fréchet Inception Distance (FID) (Heusel et al., 2017) between semantic renderings of generated and real bounding-box layouts. Compatibility is measured by graph edit

distance (GED) (Abu-Aisheh et al., 2015) between the input graph and the graph reconstructed from generated geometry. Lower FID and GED are better. Each setting uses 1,000 test layouts; we report the mean and standard deviation over five generation runs.

For a comprehensive comparison, we followed prior work and evaluated all methods under both separate and mixed training settings. Under separate training, layouts with the room count reserved for testing were excluded from training, with results reported in Table 2. Under mixed training, layouts with different room counts were combined, while the bubble graph topologies in the test set did not appear in the training set. We consider this setting closer to realistic application scenarios, where layouts of different scales are jointly available. Results under mixed training are reported in Table 1.

Table 1: Quantitative comparison under mixed training. Visual and functional realism are average preference scores. Diversity and compatibility are reported as mean/std.

Model	Visual Realism ↑	Functional Realism ↑	Diversity ↓	Compatibility ↓
HouseGAN++	-0.58	-0.61	13.9/0.3	4.0/0.0
LayoutTransformer	0.13	0.08	22.0/0.4	3.4/0.0
HouseDiffusion	0.61	0.58	12.0/0.2	1.5/0.1
Ours	**0.68**	**0.69**	**9.0/0.2**	**0.8/0.0**

Table 2: Quantitative comparison under separate training. Integers denote the number of rooms reserved for testing and excluded from training.

Model	Diversity ↓				Compatibility ↓			
	5	6	7	8	5	6	7	8
HouseGAN++	19.0/0.3	22.5/0.3	20.0/0.3	13.6/0.4	2.3/0.0	2.2/0.0	3.5/0.0	3.5/0.0
LayoutTransformer	22.8/0.2	18.3/0.4	18.3/0.2	30.6/0.7	0.7/0.0	2.3/0.0	3.2/0.0	4.3/0.1
HouseDiffusion	10.3/0.2	12.0/0.3	12.9/0.3	13.5/0.1	0.9/0.0	1.2/0.0	1.4/0.0	2.0/0.0
Ours	**9.8/0.3**	**8.2/0.1**	**8.8/0.2**	**12.3/0.5**	**0.3/0.0**	**0.4/0.0**	**0.8/0.0**	**1.7/0.0**

4.3 Ablation Study

Table 3: Ablation study. (a) Proposed components under mixed and 8-room settings. Base denotes the hybrid-order autoregressive baseline of Liu et al. (2022) without graph conditioning; FO, PGR, and GAR denote Functional Order, Pretrained Graph Representation, and Graph-Aligned Representation, respectively. (b) PGP, JGM, and PGR denote partial graph propagation, joint graph memory, and pretrained graph representation. (c) compares no graph alignment, category-shared alignment, and node-aligned representation on the 8-room setting. Diversity and compatibility are reported as mean/std.

(a) Proposed components

Components				Diversity ↓		Compatibility ↓	
Base	FO	PGR	GAR	Mixed	8	Mixed	8
✓				22.3/0.2	24.0/0.9	2.6/0.0	3.8/0.0
✓	✓			12.3/0.3	14.4/0.5	2.7/0.0	3.3/0.1
✓		✓		23.3/0.2	17.8/0.7	1.4/0.0	2.5/0.0
✓	✓	✓		**8.8/0.1**	13.7/0.5	1.0/0.0	1.9/0.0
✓	✓	✓	✓	9.0/0.2	**12.3/0.5**	**0.8/0.0**	**1.7/0.0**

(b) Topology conditioning

Architecture	Diversity ↓ 8	Compatibility ↓ 8
PGP	15.3/0.5	2.9/0.0
JGM	16.3/0.6	2.5/0.0
PGR	**13.7/0.5**	**1.9/0.0**

(c) Graph alignment

GAR Form	Diversity ↓ 8	Compatibility ↓ 8
No GAR	13.7/0.5	1.9/0.0
Category-shared	12.8/0.4	1.8/0.0
Node-aligned	**12.3/0.5**	**1.7/0.0**

Proposed components. We first evaluate the contribution of Functional Order (FO), Pretrained Graph Representation (PGR), and Graph-Aligned Representation (GAR). The base model adopts

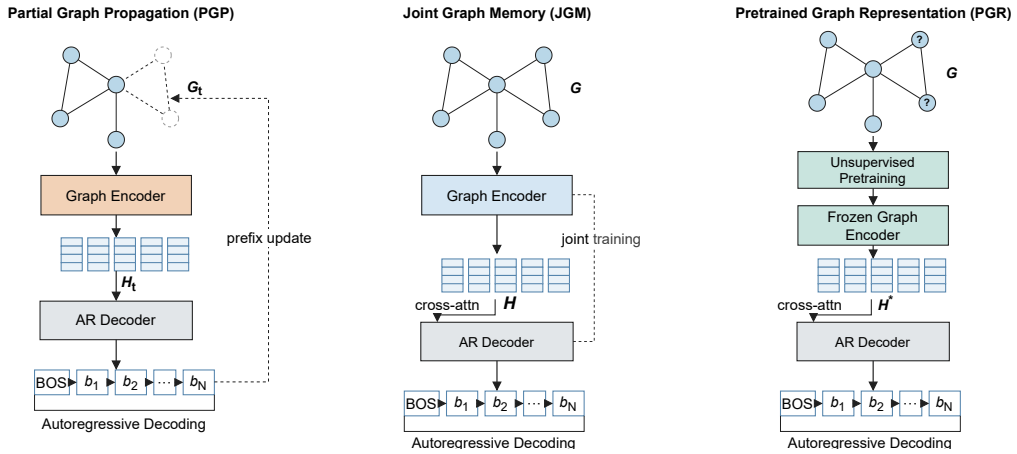


Figure 3: Topology conditioning architectures in Table 3(b). PGP updates the partial graph representation with the generated prefix. JGM encodes the complete graph as a memory jointly learned with the decoder. PGR independently pretrains and freezes the graph encoder before using its complete-graph representations throughout decoding.

the hybrid sorting strategy of Liu et al. (2022) as its generation order without additional graph conditioning. FO replaces this category-level schedule with a graph-derived node-level order, PGR provides representations learned from the complete bubble graph as conditions for autoregressive decoding, and GAR further associates each generated object with its corresponding graph node. The results in Table 3(a) show that FO mainly improves layout diversity, while PGR substantially improves topology compatibility. Their combination achieves stronger overall performance, indicating that sequence organization and graph representation provide complementary benefits for autoregressive generation. Adding GAR further improves compatibility and produces clearer gains on the 8-room setting, where longer sequences and more complex graph relations increase the need for node-specific information.

Topology conditioning architecture. Figure 3 contrasts three ways of introducing graph structure into autoregressive generation, evaluated in Table 3(b). All variants use FO as the generation order and exclude GAR. For a controlled comparison, they employ the same two-layer GAT architecture for graph encoding. *Partial Graph Propagation (PGP)* applies the GAT to the graph available at each autoregressive step, allowing topology information to evolve together with the generated sequence. *Joint Graph Memory (JGM)* encodes the complete input graph before decoding and provides the resulting node features through cross-attention, with the graph encoder and geometry generator optimized jointly. *Pretrained Graph Representation (PGR)* uses the same complete-graph conditioning architecture, while learning the graph representation through masked graph pretraining before geometry generation and keeping the encoder fixed during downstream training. JGM improves compatibility by making the complete input structure available throughout decoding, while PGR achieves the best FID and GED. These results show that complete graph conditioning benefits topology compatibility, and independently pretrained topology representations provide the strongest overall condition for autoregressive decoding.

Graph alignment granularity. Finally, we examine how precisely graph representations should be aligned with generated objects. The analysis is conducted on the 8-room setting, where repeated room categories make individual node distinction more relevant. Without GAR, the decoder relies on global graph conditioning without an explicit node-specific representation. In the *Category-shared* variant, nodes of the same semantic category receive their averaged graph representation, while *Node-aligned* preserves the representation of the exact graph node corresponding to each generated object. Category-shared alignment improves FID from 13.7 to 12.8 and GED from 1.9 to 1.8, while node-level alignment further improves them to 12.3 and 1.7. The consistent improvement indicates that category-level graph information does not fully capture the different topology roles of nodes with the same semantic type, and preserving node-specific correspondence provides more precise structural guidance for geometry generation.

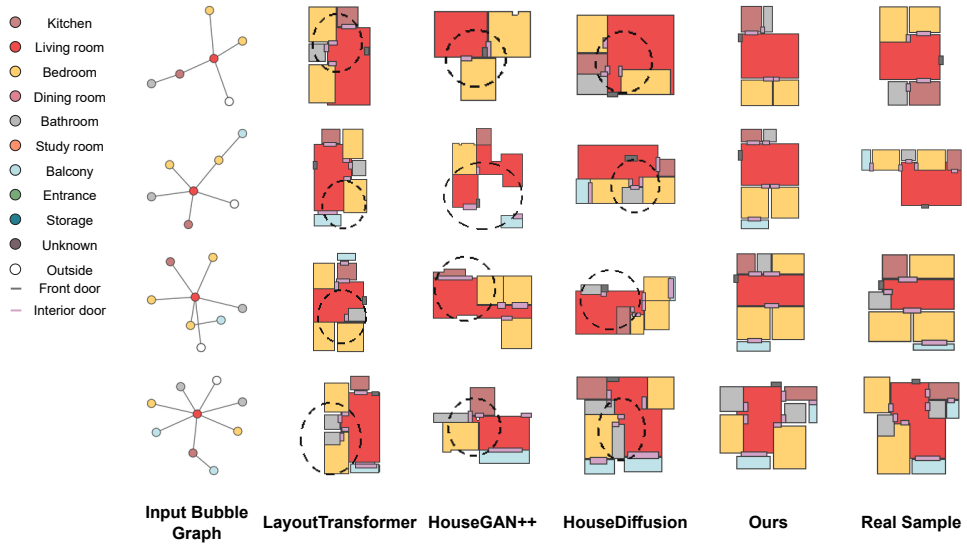


Figure 4: Qualitative comparisons on fixed mixed-test examples. All generated layouts and real samples are rendered using the same room and door palette.

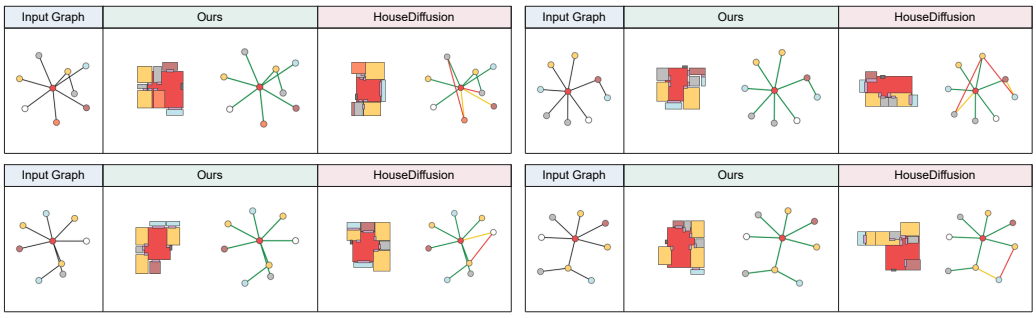


Figure 5: Connectivity comparisons between our method and HouseDiffusion. For each method, the generated layout is followed by the graph reconstructed from it. Green, yellow, and red edges indicate correct, missing, and extraneous connections, respectively.

4.4 Result Analysis

Under mixed training, our method achieves the best scores on visual realism, functional realism, FID, and GED. Compared with HouseDiffusion, the strongest baseline, it reduces FID from 12.0 to 9.0 and GED from 1.5 to 0.8. The fixed-case samples in Figure 4 complement these aggregate scores by showing how different methods realize the same bubble diagram as room and door geometry under a common rendering style. Figure 5 makes the topology difference more explicit by placing each generated layout beside its reconstructed graph: missing and extraneous connections expose errors that visual plausibility alone can conceal.

Table 2 reports the results under separate training, where the room count used for testing is excluded from training. Our method obtains the best FID and GED across all four splits. The advantage over HouseDiffusion persists as the held-out room count increases, although compatibility worsens for both methods on larger layouts. In the 5–7-room splits, our mean GED remains below 1.0. Together with the mixed-training results, this suggests that complete-graph conditioning helps preserve connectivity even when the test room count is unseen during training.

5 DISCUSSION

The mixed and unseen room-count results indicate that complete-graph representations carry structural information beyond individual training layouts. The ablations clarify their two roles: all node representations supply floorplan-level context, while the aligned node representation identifies the current object. FO makes these roles compatible with ordered coordinate prediction.

Pretraining and freezing the graph encoder separates topology learning from the changing geometry prefix, while alignment retains object identity during decoding. This conditioning adds computation; the supplementary material reports inference speed and memory alongside the baselines.

The current evaluation covers at most eight rooms, and topology compatibility declines as layouts grow. Longer sequences and richer conditions remain important extensions: real design briefs also specify circulation, hierarchy, dimensions, and building boundaries, potentially through text, sketches, or reference images.

6 CONCLUSION

We propose an autoregressive framework that learns representations from the complete input graph before generating vector floorplans. FO aligns graph nodes with the output sequence; the same representations condition the full decoder and identify individual objects through GAR. On RPLAN, this combination improves realism, diversity, and topology compatibility across the evaluated mixed and separate settings.

AI USE STATEMENT

We used generative AI tools to assist with language editing and LaTeX formatting. Generative AI tools were not used to generate experimental data or to determine the methods, results, or scientific claims reported in this paper. All AI-assisted text and formatting changes were reviewed by the authors, who take responsibility for the final content of this work.

ETHICS STATEMENT

This work uses the publicly available RPLAN dataset and studies floorplan generation as a design-support task. The preference study asks participants to assess generated layouts and reports only aggregated scores. The generated floorplans are research outputs rather than construction-ready designs and should be reviewed by qualified designers before practical use.

REPRODUCIBILITY STATEMENT

Sections 3 and 4 describe the model, preprocessing, evaluation metrics, data splits, and repeated generation protocol used in our experiments. The supplementary material provides detailed architecture configurations, efficiency comparisons, and the user-study protocol.

REFERENCES

Zeina Abu-Aisheh, Romain Raveaux, Jean-Yves Ramel, and Patrick Martineau. An exact graph edit distance algorithm for solving pattern recognition problems. In *Proceedings of the International Conference on Pattern Recognition Applications and Methods – Volume 1*, pp. 271–278. SciTePress, 2015. doi: 10.5220/0005209202710278.

Diego Martin Arroyo, Janis Postels, and Federico Tombari. Variational transformer networks for layout generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13642–13652, 2021.

Fan Bao, Dong-Ming Yan, Niloy J. Mitra, and Peter Wonka. Generating and exploring good building layouts. *ACM Transactions on Graphics*, 32(4):Article 122, 2013. doi: 10.1145/2461912.2461977.

Sumit Bisht, Krishnendra Shekhawat, Nitant Upasani, Rahil N. Jain, Riddhesh Jayesh Tiwaskar, and Chinmay Hebbar. Transforming an adjacency graph into dimensioned floorplan layouts. *Computer Graphics Forum*, 41(6):5–22, 2022. doi: 10.1111/cgf.14451.

Jiacheng Chen, Chen Liu, Jiaye Wu, and Yasutaka Furukawa. Floor-SP: Inverse CAD for floorplans by sequential room-wise shortest path. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2661–2670, 2019.

Steve Cruz, Will Hutchcroft, Yuguang Li, Naji Khosravan, Ivaylo Boyadzhiev, and Sing Bing Kang. Zillow indoor dataset: Annotated floor plans with 360deg panoramas and 3d room layouts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2133–2143, 2021.

Mohammed Haroon Dupty, Yanfei Dong, Sicong Leng, Guoji Fu, Yong Liang Goh, Wei Lu, and Wee Sun Lee. Constrained layout generation with factor graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12851–12860, 2024. doi: 10.1109/CVPR52733.2024.01221.

Zhiwen Fan, Lingjie Zhu, Honghua Li, Xiaohao Chen, Siyu Zhu, and Ping Tan. FloorPlanCAD: A large-scale CAD drawing dataset for panoptic symbol spotting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10128–10137, 2021. doi: 10.1109/ICCV48922.2021.00997.

Kamal Gupta, Justin Lazarow, Alessandro Achille, Larry S. Davis, Vijay Mahadevan, and Abhinav Shrivastava. LayoutTransformer: Layout generation and completion with self-attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1004–1014, 2021.

Feixiang He, Yanlong Huang, and He Wang. iPLAN: Interactive and procedural layout planning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7793–7802, 2022.

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, 2017.

Shibo Hong, Xuhong Zhang, Tianyu Du, Sheng Cheng, Xun Wang, and Jianwei Yin. Cons2Plan: Vector floorplan generation from various conditions via a learning framework based on conditional diffusion models. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pp. 3248–3256, 2024. doi: 10.1145/3664647.3680681.

Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. GraphMAE: Self-supervised masked graph autoencoders. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 594–604. ACM, 2022. doi: 10.1145/3534678.3539321.

Zhenyu Hou, Yufei He, Yukuo Cen, Xiao Liu, Yuxiao Dong, Evgeny Kharlamov, and Jie Tang. GraphMAE2: A decoding-enhanced masked self-supervised graph learner. In *Proceedings of the ACM Web Conference 2023*, pp. 737–746. ACM, 2023. doi: 10.1145/3543507.3583379.

Ruizhen Hu, Zeyu Huang, Yuhan Tang, Oliver Van Kaick, Hao Zhang, and Hui Huang. Graph2Plan: Learning floorplan generation from layout graphs. *ACM Transactions on Graphics*, 39(4):118:1–118:14, 2020.

Sizhe Hu, Wenming Wu, Yuntao Wang, Benzhu Xu, and Liping Zheng. GSDiff: Synthesizing vector floorplans via geometry-enhanced structural graph generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 17323–17332, 2025.

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

Akash Abdu Jyothi, Thibaut Durand, Jiawei He, Leonid Sigal, and Greg Mori. LayoutVAE: Stochastic scene layout generation from a label set. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9895–9904, 2019.

Ahti Kalervo, Juha Ylioinas, Markus Häikiö, Antti Karhu, and Juho Kannala. CubiCasa5K: A dataset and an improved multi-task model for floorplan image analysis. In *Scandinavian Conference on Image Analysis*, pp. 28–40, 2019. doi: 10.1007/978-3-030-20205-7_3.

Xiang Kong, Lu Jiang, Huiwen Chang, Han Zhang, Yuan Hao, Haifeng Gong, and Irfan Essa. BLT: Bidirectional layout transformer for controllable layout generation. In *European Conference on Computer Vision*, pp. 474–490, 2022.

Jiachen Liu, Yuan Xue, Jose Duarte, Krishnendra Shekhawat, Zihan Zhou, and Xiaolei Huang. End-to-end graph-constrained vectorized floorplan generation with panoptic refinement. In *European Conference on Computer Vision*, volume 13675, pp. 547–562, 2022. doi: 10.1007/978-3-031-19784-0_32.

Charlie Nash, Yaroslav Ganin, S. M. Ali Eslami, and Peter Battaglia. PolyGen: An autoregressive generative model of 3d meshes. In *International Conference on Machine Learning*, pp. 7220–7229. PMLR, 2020.

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 *European Conference on Computer Vision*, pp. 162–177, 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*, pp. 13632–13641, 2021.

Wamiq Para, Paul Guerrero, Tom Kelly, Leonidas J. Guibas, and Peter Wonka. Generative layout modeling using constraint graphs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6690–6700, 2021.

Mohammad Amin 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*, pp. 5466–5475, 2023.

Kaede Shiohara and Toshihiko Yamasaki. Unified vector floorplan generation via markup representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 39262–39271, 2026.

Jiahui Sun, Wenming Wu, Ligang Liu, Wenjie Min, Gaofeng Zhang, and Liping Zheng. WallPlan: Synthesizing floorplans by learning to generate wall graphs. *ACM Transactions on Graphics*, 41 (4):Article 92, 2022. doi: 10.1145/3528223.3530135.

Hao Tang, Zhenyu Zhang, Humphrey Shi, Bo Li, Ling Shao, Nicu Sebe, Radu Timofte, and Luc Van Gool. Graph transformer GANs for graph-constrained house generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2173–2182, 2023.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pp. 5998–6008, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Shidong Wang and Renato Pajarola. Eliminating rasterization: Direct vector floor plan generation with DiffPlanner. *IEEE Transactions on Visualization and Computer Graphics*, 31(10):7906–7922, 2025. doi: 10.1109/TVCG.2025.3559682.

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*, 38(6):Article 234, 2019. doi: 10.1145/3355089.3356556.

Hang Zhang, Anton Savov, and Benjamin Dillenburger. MaskPLAN: Masked generative layout planning from partial input. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8964–8973, 2024.