Supplementary Material

A. Additional Conditional Generation Results

We present additional conditional generation examples using our proposed model in Figure 1. Each cell shows a masked semantic graph as input (left) and the corresponding 3D furnished layout generated by Nestwork (right). Additional generated layouts are shown in Figure 2. These examples demonstrate the model's ability to interpret semantic structure and produce diverse, realistic spatial arrangements.

B. Heterogeneous Graph Construction

B.1. Dataset Preprocessing

Data Filtering. We begin by preprocessing the 3D-FRONT dataset following the ATISS [5] script to filter out unreasonable house data: *(1)* Rooms with zero dimensions are removed. *(2)* Rooms such as living rooms without furniture are excluded. *(3)* Rooms of excessive size are filtered out.

At the house level, we apply custom filtering functions: *(1)* Houses must contain basic rooms (bedrooms and bathrooms). *(2)* Houses with disconnected rooms are removed (i.e., the room graph must be connected). *(3)* Houses with fewer than two rooms are excluded.

Edge Connectivity Detection. After filtering, we detect edge connectivity for each house:

- **Furniture-Furniture**: The furniture subgraph is fully connected, with a maximum of 15 furniture items allowed per room.
- **Furniture-Room and Room-Furniture**: These bidirectional edges always connect each furniture node to its parent room.
- **Room-House and House-Room**: These bidirectional edges always connect each room node to the house node.
- **Room-Room**: Edges are detected based on geometric distance with a threshold. We define two types of room-room edges:
  - *Adjacent Rooms*: Two rooms are considered adjacent if, in the xz-plane, their normalized boxes either overlap or are separated by a small gap (at most 0.06 of the unit size) along x or z while overlapping along the other axis. Intuitively, rooms that touch or are very close along a wall are marked as adjacent.
  - *Far Rooms*: Two rooms are considered far if their normalized separation along x or z is large (at least 0.3 of the house size), and they are not already marked as adjacent. In practice, we compute the minimal gap along x and z, take the larger of the two, and if this value is ≥ 0.3 the pair is labeled as a far-room edge.

B.2. Node Features

The node features in our heterogeneous graph are designed to capture the geometric and semantic properties. They are composed of four components:

Bounding box parameters. Each node is associated with its bounding box parameters, represented as $[dx, dy, dz, minx, miny, minz]$ where: dx, dy, dz: Dimensions of the bounding box along the x, y, and z axes. $minx, miny, minz$: The minimum corner coordinates of the bounding box.

These parameters are normalized by the size of the parent bounding box to ensure that nodes within the same hierarchical structure (e.g., furniture within a room or rooms within a house) share a consistent local coordinate system. For the house-level node, absolute dimensions are used, and the location is set to $[0, 0, 0]$.

Angles. For furniture nodes, angles are discretized into categorical data. First, the rotation angle in degrees is extracted. Then, using a predefined maximum number of angle categories (we use 4 in our experiments), the angle is discretized into one of several bins. This approach ensures that angular information is represented in a format suitable for categorical embeddings. For room and house nodes, the angles are set to 0.

Class labels. Each node is assigned a class label that represents its semantic category. We use two sets of class labels:

(1) The full furniture category set contains fine-grained furniture categories such as `"chaise_longue_sofa"` and `"l_shaped_sofa"`

(2) A simplified version that maps furniture categories to broader groups. For instance, `"chaise_longue_sofa"` and `"l_shaped_sofa"` are mapped to `"sofa"`.

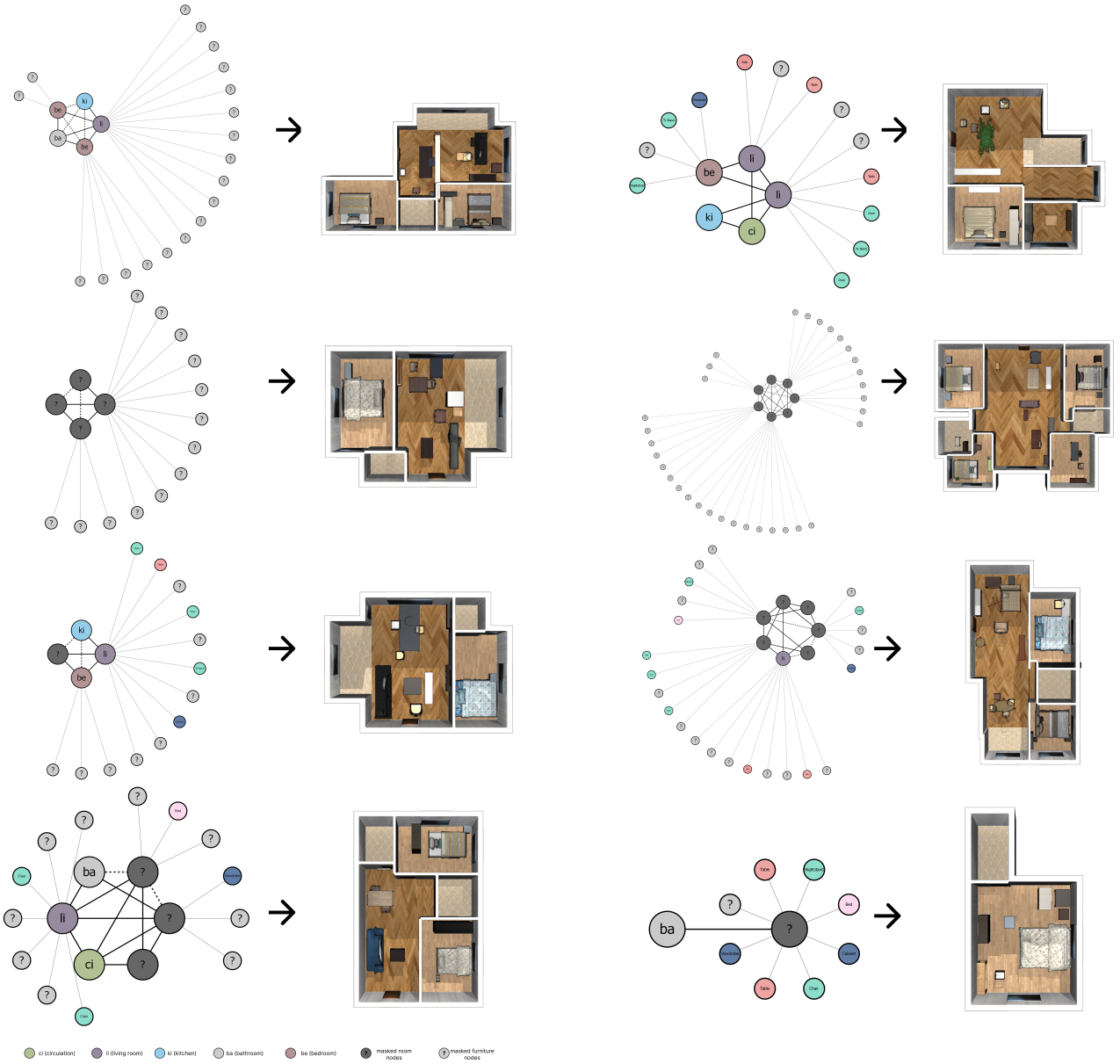

Figure 1. Additional conditional generation examples. Each layout is conditioned on a masked semantic graph and generated with Nestwork.

Both furniture and room nodes share the same class label system. The labels are encoded using one-hot embeddings. In our experiments, we use the simplified version of furniture categories.

Shape Features. Following the protocol described in CommonScenes [9], we use DeepSDF to encode each furniture mesh into a 256-dimensional latent vector. This representation captures detailed geometric information about the shape of each furniture item and provides a compact yet expressive feature for downstream tasks. In room nodes, the shape features are initialized from category and bounding box embeddings.

B.3. Edge Features

Edge features between bounding boxes $\mathcal{B}_1$ and $\mathcal{B}_2$ form $\mathbf{f} \in \mathbb{R}^{12}$. For box $i \in \{1, 2\}$, let

$$\mathbf{p}_i^{\min} = [p_{i,x}^{\min}, p_{i,y}^{\min}, p_{i,z}^{\min}], \qquad \mathbf{p}_i^{\max} = [p_{i,x}^{\max}, p_{i,y}^{\max}, p_{i,z}^{\max}]$$

Figure 2. Additional generated residential layouts from Nestwork

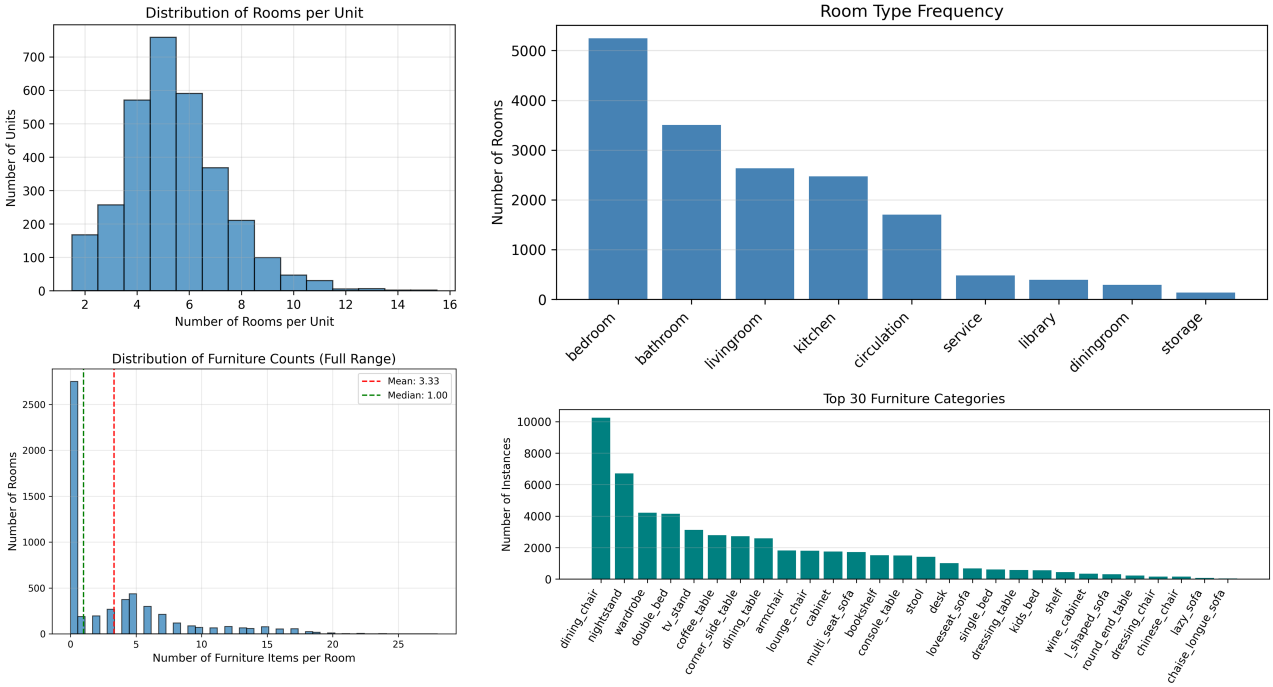

Figure 3. Room and furniture statistics of HG-FRONT dataset

denote the minimum and maximum corners of $\mathcal{B}_i$. We define the centroid and box size as

$$\mathbf{c}_i = \frac{\mathbf{p}_i^{\min} + \mathbf{p}_i^{\max}}{2} = [c_{i,x}, c_{i,y}, c_{i,z}],$$

$$\mathbf{d}_i = \mathbf{p}_i^{\max} - \mathbf{p}_i^{\min} = [d_{i,x}, d_{i,y}, d_{i,z}],$$

with volume

$$V_i = \prod_{j \in \{x,y,z\}} d_{i,j},$$

and yaw angle θ_i about the z-axis. Let the parent-box size be

$$\mathbf{s}_p = [s_{p,x}, s_{p,y}, s_{p,z}],$$

where each component is lower-bounded by $\epsilon = 10^{-8}$, i.e.,

$$s_{p,j} = \max(s_{p,j}^{\text{raw}}, \epsilon), \qquad j \in \{x, y, z\}.$$

Centroid distance (3D).

$$\mathbf{f}_{\text{cen}} = \frac{\mathbf{c}_1 - \mathbf{c}_2}{\mathbf{s}_p}$$

Signed bbox distance (3D). For each axis $j \in \{x, y, z\}$:

$$\delta_j = \min(p_{1,j}^{\max}, p_{2,j}^{\max}) - \max(p_{1,j}^{\min}, p_{2,j}^{\min})$$

$$f_{\text{bbox},j} = \begin{cases} -\delta_j / s_{p,j} & \text{if } |\delta_j| > \epsilon \\ 0 & \text{if } |\delta_j| \le \epsilon \end{cases}$$

where $\delta_j > 0$ indicates overlap (hence negative signed distance), $\delta_j = 0$ indicates contact, and $\delta_j < 0$ indicates separation (positive signed distance). Collecting the three components gives $\mathbf{f}_{\text{bbox}} = [f_{\text{bbox},x}, f_{\text{bbox},y}, f_{\text{bbox},z}]$.

Angle difference (1D).

$$f_{\text{angle}} = ((\theta_1 - \theta_2 + \pi) \bmod 2\pi) - \pi \in [-\pi, \pi]$$

Dimension difference (3D).

$$\mathbf{f}_{\text{dim}} = \frac{\mathbf{d}_1 - \mathbf{d}_2}{\mathbf{s}_p}$$

IoU (1D).

$$\mathbf{p}_{\cap}^{\min} = \max(\mathbf{p}_1^{\min}, \mathbf{p}_2^{\min})$$

$$\mathbf{p}_{\cap}^{\max} = \min(\mathbf{p}_1^{\max}, \mathbf{p}_2^{\max})$$

$$\mathbf{d}_{\cap} = \max(\mathbf{0}, \mathbf{p}_{\cap}^{\max} - \mathbf{p}_{\cap}^{\min})$$

$$V_{\cap} = \prod_{j \in \{x,y,z\}} d_{\cap,j}$$

$$f_{\text{iou}} = \begin{cases} 0 & \text{if } \exists j : d_{\cap,j} \le 0 \\ \frac{V_{\cap}}{V_1 + V_2 - V_{\cap}} & \text{otherwise} \end{cases}$$

Thus, face or edge contact yields zero IoU.

Log volume ratio (1D).

$$f_{\text{vol}} = \begin{cases} 0 & \text{if } V_1 = V_2 = 0 \\ -10 & \text{if } V_1 = 0, V_2 > 0 \\ 10 & \text{if } V_1 > 0, V_2 = 0 \\ \text{clip}(\ln V_1 - \ln V_2, -10, 10) & \text{otherwise} \end{cases}$$

where $\text{clip}(x, a, b) = \max(a, \min(b, x))$.

Feature vector.

$$\mathbf{f} = [\mathbf{f}_{\text{cen}}; \mathbf{f}_{\text{bbox}}; f_{\text{angle}}; \mathbf{f}_{\text{dim}}; f_{\text{iou}}; f_{\text{vol}}] \in \mathbb{R}^{12}$$

Dimensions: $3 + 3 + 1 + 3 + 1 + 1 = 12$. Normalization by $\mathbf{s}_p$ ensures scale invariance; log volume ratio and angle wrapping improve numerical stability.

B.4. Room/Furniture Statistics

Our filtered HG-FRONT subset contains 3,111 valid residential units with an average of 5.4 rooms per unit (Fig. 3, top left), dominated by program types such as bedrooms, bathrooms, living rooms, and kitchens (top right). Across 16,858 rooms, furniture counts follow a heavy-tailed distribution; the empirical maximum reaches 20 (bottom left), but cases above 15 lie in a very thin tail. We therefore cap each room at 15 furniture nodes, which covers nearly all examples while providing a stable and well-behaved upper bound for heterogeneous graph construction.

B.5. Dataset Limitations.

The HG-FRONT data inherits several structural limitations from 3D-FRONT. First, kitchens and bathrooms are consistently exported as *empty* rooms–no appliances, fixtures, or storage elements are provided–so their furniture counts are always zero. Second, *door annotations are missing or inconsistent across units*, which precludes their reliable use in the graph. Accordingly, we omit doors from the training graph and add doors and windows only as a lightweight post-processing step for visualization. Finally, the source layouts themselves contain non-zero object overlap and tight-clearance cases, as also noted in prior work on 3D-FRONT [7], so under our evaluation protocol the HG-FRONT ground-truth collision baseline is non-zero (9.8%). These limitations reflect the underlying data rather than model behavior.

C. Implementation Details

C.1. Main Model Implementation Details

Heterogeneous Autoencoder. Each node attribute has separate projections (e.g., bounding box, category, shape features). These embeddings are concatenated into a 128-D node feature. The encoder is a 3-layer HetGAT with 4 heads per layer (128-D node and edge features, 64-D edge-type embeddings), producing a 128-D latent per node. The decoder mirrors this architecture and reconstructs all node attributes. This autoencoder establishes the latent space on which the diffusion prior operates.

Algorithm 1 One Denoising Layer in Latent DDPM

Require: Node features x, timestep embedding T, masked semantics C, graph-level conditioning G, edges E

$x \leftarrow x + \text{Proj}_t(T)$
// self-attention block
$h \leftarrow \text{HetGAT}(x, E)$
$x \leftarrow \text{LayerNorm}\big(\text{Proj}(x) + \text{FFN}(h)\big)$
// compute residuals from node- and graph-level conditioning
$\text{res} \leftarrow W_{\text{node}}\, C$
if G exists **then**
  $\text{res} \leftarrow \text{res} + W_{\text{graph}}\, G$
end if
// masked cross-attention
$Q \leftarrow W_q x$
$K \leftarrow W_k C$
$V \leftarrow W_v C$
$m \leftarrow \text{CrossAttn}(Q, K, V, E)$
$x \leftarrow x + \alpha_{\text{cond}}\,(m + \text{res})$
$x \leftarrow \text{LayerNorm}\big(x + \text{FFN}_{\text{post}}(x)\big)$
return x

Latent Diffusion Prior. We train a DDPM over the heterogeneous latent graph. At timestep t, noisy latents $\mathbf{z}_t \in \mathbb{R}^{128}$ are projected to 256-D and passed through a 5-layer HetGAT denoiser. Each layer performs: (1) timestep embedding, (2) heterogeneous self-attention, (3) masked cross-attention driven by the semantic graph, and (4) optional graph-level conditioning. The detailed logic of one denoising layer is shown in Algorithm 1. Graph-level text conditioning is an optional interface branch, disabled in all main-paper experiments and used only for LLM text-to-graph, where the LLM parser also emits the structured prompt expected by this branch.

After five denoising layers, the final hidden representation $X^{(5)} \in \mathbb{R}^{N \times 256}$ is projected through W_{out} to produce $\hat{X}_0 \in \mathbb{R}^{N \times 128}$, the DDPM prediction of the clean latent graph under a 200-step noise schedule.

Structured Prompt for the Optional Text Branch. When this branch is enabled, we represent each masked graph as a compact structured prompt. During training, this prompt is obtained by serializing the masked semantic graph; in the LLM text-to-graph interface, the parser emits the same template alongside the graph.

Each node (room or furniture) is converted into a short textual token based on its type and mask state. For example:

- **All furniture masked:** A living room with no visible furniture becomes *"a living room with unknown furniture"*.
- **Partially masked furniture:** A bedroom containing a visible bed but masked nightstands becomes *"a bedroom containing a bed; other furniture unknown"*.
- **Multi-room description:** A layout with several rooms yields a concatenated prompt such as *"a living room with sofa and table; a kitchen with unknown furniture; one additional unspecified room"*.

The structured prompt is encoded using a frozen BERT-base encoder (768-D) and passed through an 8-head adapter to produce graph-level conditioning vectors for the optional text branch.

C.2. Two-Stage Baseline Implementation Details

Stage 1: HouseDiffusion for Room Layouts. We retrain HouseDiffusion [6] from scratch on the HG-FRONT dataset to serve as the first-stage room-layout generator. The core architecture, tokenization scheme, and continuous denoising objective follow the original implementation without modification. Because HG-FRONT does not provide door geometry, we adapt the relational cross-attention to use *room–room adjacency* rather than room–door connectivity when constructing the attention mask.

Prior to tokenization, every floorplan is normalized into a 255×255 square via an aspect-ratio-preserving affine transform; the corresponding inverse transform is stored so that predicted polygons can be mapped back to world coordinates at inference time. The model is trained for 250k steps on the HG-FRONT split.

Stage 2: DiffuScene for Room-Level Furniture Layout. For interior synthesis, we retrain DiffuScene [8] on the HG-FRONT dataset following its official training pipeline. We train it on room-mask conditioning mode and preserve the original strategy for furniture-count control, where the model selects the top-K most probable furniture categories during sampling. Because DiffuScene operates at the room level, we apply it independently to each room polygon generated by HouseDiffusion and then merge all predicted furniture back into the global coordinate frame, without altering the original network architecture.

Overall Pipeline. At inference time, HouseDiffusion receives the room-room adjacency graph and produces all room polygons, which are mapped back to world coordinates using the stored inverse transform. Each predicted room is then passed to DiffuScene, which generates candidate furniture items conditioned on the room mask; we keep the top-K samples following the original DiffuScene procedure. This two-stage process is effectively equivalent to our *room-graph evaluation mode*, where the input graph contains the rooms but all furniture nodes are masked: because the furniture subgraph in Nestwork forms a clique, masking it removes relational constraints and reduces conditioning to a furniture-count signal–mirroring the behavior of the two-stage baseline.

C.3. Autoencoder Backbone Variants

For the autoencoder ablation in Section 4.5 (Autoencoder Backbone Study), we keep all components of Nestwork fixed–denoising backbone, LRF module, heterogeneous node/edge types, and training setup–and replace only the GNN backbone used for the encoder and decoder. We evaluate two baselines from existing 3D scene-graph literature: TripletGCN and SLN-Box.

TripletGCN. TripletGCN is adapted from the graph-to-3D [2] used in scene-graph conditioned layout generation. We apply five homogeneous Triplet-GCN layers to our heterogeneous house graphs for both encoder and decoder. Each layer processes rooms and furniture through shared message-passing, without type-specific parameters, and encodes into a 128-D latent vector to match the latent dimension used in our main model. This backbone is designed to capture geometric relations among objects, but lacks the node-type-aware attention used in Nestwork.

SLN-Box. SLN-Box follows the structure of the original 3D-SLN model [4], using five plain GCN layers without residual projection sharing. Only box attributes are reconstructed from a 64-D latent, following the original design that targets bounding box prediction without furniture shape features. During inference, furniture meshes are retrieved via nearest-neighbor search over box parameters, exactly as in the original pipeline.

Adaptation to Heterogeneous Graph. Both TripletGCN and SLN-Box were originally designed for room-level scene graphs with richer, manually defined edge semantics (e.g., *left-of*, *right-of*). Compared to original implementations in [2, 4], the adaptation to our heterogeneous graph data includes:

1. **Encoder:** we replace symbolic geometric edge labels with numeric edge features (relative position, distance, orientation, etc.).
2. **Decoder:** we use purely categorical edge types (e.g., *room-furniture*, *furniture-furniture*) instead of symbolic labeled edges. The geometric information that would otherwise be encoded in edge labels is supplied by the LRF module in the decoder.

These adaptations allow both baselines to operate on the same heterogeneous graph structure as Nestwork while preserving their architectural characteristics. The comparison

therefore isolates the impact of backbone expressiveness on downstream diffusion performance.

C.4. Latent-prior Implementations

This section details the two latent prior baselines evaluated in Section 4.6 (Ablation on Latent Priors).

Autoregressive Prior. We implement an autoregressive prior following the published method description in Chattopadhyay et al. [1] as an alternative prior over furniture latents. Each room is assigned a maximum of $K{=}15$ latent slots. We first process the semantic room-furniture graph with a lightweight GNN to obtain per-room embeddings; these are concatenated with the corresponding room latent and used to initialise the hidden state of a GRU. The GRU then autoregressively predicts a Gaussian distribution at each step,

$$P(\mathbf{z}_t \mid \mathbf{z}_{<t}) = \mathcal{N}(\mu_t, \sigma_t^2),$$

and samples $\mathbf{z}_t = \mu_t + \sigma_t \odot \varepsilon$. A full sequence of K latents is generated, but only the first m samples are kept, where m is the true furniture count specified by the semantic graph.

Because furniture nodes have no canonical ordering, we apply Hungarian matching [3] between the m predicted latents and the m ground-truth latents before computing the KL loss. This is identical to the matching strategy used in Chattopadhyay et al. [1], and remains fully differentiable since gradients pass through the KL terms after the permutation alignment. The decoder and encoder architectures are unchanged, isolating the effect of the autoregressive prior alone.

IID Prior. As a second baseline, we remove the learnable prior entirely and sample every latent furniture node independently from the standard normal distribution $\mathcal{N}(0, I)$ at inference. The decoder is identical to that of Nestwork, so this baseline evaluates how well the autoencoder itself can generate plausible layouts without any structured latent prior.

D. Additional Experiments and Analysis

D.1. Denoising Backbone Architecture

Our main model employs a heterogeneous graph attention layer (HetGAT) as the self-attention module inside the denoising backbone. This choice allows the diffusion network to treat room and furniture nodes with distinct relational semantics, preserving heterogeneity during iterative refinement. To assess the importance of this design, we replace HetGAT with three homogeneous alternatives commonly used in graph generative models: a homogeneous GAT, a homogeneous Graph Transformer, and a homogeneous GCN layer. All other components, including the latent autoencoder and diffusion schedule, remain unchanged. We report all metrics under full semantic graph conditioning.

Discussion. Homogeneous architectures achieve competitive FID/KID but collapse in graph-constraint accuracy, dropping from over 92% (HetGAT) to roughly 63-65%. This indicates that while global visual appearance can be recovered with generic message passing, correctly preserving fine-grained relational structure between heterogeneous node types requires type-aware attention. The HetGAT backbone therefore plays a critical role in maintaining semantic validity during iterative denoising, justifying its use in the main model.

D.2. LRF Number of Slots

Setup. We study how the number of relational slots K in the LRF dictionary affects generation quality. We keep all other components fixed and vary only $K \in \{4, 6, 8\}$ under full semantic graph conditioning. We report FID/KID, graph-constraint accuracy, collision rate, and diversity metrics (size, location, angle).

Discussion. The results show that a moderate number of slots provides the best overall trade-off. Reducing the dictionary to $K{=}4$ weakens graph satisfaction and increases collisions, indicating insufficient relational capacity. Increasing to $K{=}8$ does not improve fidelity and slightly reduces both diversity and graph accuracy, suggesting diminishing returns from a larger slot vocabulary. We therefore adopt $K{=}6$ as the default setting.

D.3. LRF Rank Size

Setup. We study how the low-rank dimension r in the LRF module affects generation quality. We keep all other components fixed and vary only $r \in \{16, 32, 64\}$ under full semantic graph conditioning. We report FID/KID, graph-constraint accuracy, collision rate, and diversity metrics (size, location, angle).

Discussion. The results show that a moderate rank provides the best trade-off between fidelity and structural validity. Reducing the rank to $r{=}16$ limits relational capacity and degrades both FID/KID and graph-constraint accuracy. Increasing the rank to $r{=}64$ does not improve performance and slightly worsens collisions and graph satisfaction, suggesting diminishing returns and weaker regularization. We therefore adopt $r{=}32$ as the default setting.

Denoising Backbone	FID ↓	KID ↓ ($\times 100$)	Coll. % ↓	Graph % ↑
HetGAT (ours)	$7.13_{\pm 0.14}$	$0.29_{\pm 0.01}$	$11.21_{\pm 0.19}$	$\mathbf{92.34}_{\pm 0.14}$
Homogeneous GAT	$8.29_{\pm 0.15}$	$0.41_{\pm 0.01}$	$12.53_{\pm 0.26}$	$62.61_{\pm 0.88}$
Homogeneous Graph Transformer	$\mathbf{6.85}_{\pm 0.12}$	$\mathbf{0.28}_{\pm 0.02}$	$\mathbf{10.90}_{\pm 0.33}$	$64.73_{\pm 0.29}$
Homogeneous GCN	$11.29_{\pm 0.15}$	$0.58_{\pm 0.01}$	$15.68_{\pm 0.32}$	$63.32_{\pm 0.24}$

Table 1. **Ablation of denoising backbone architectures under full semantic conditioning.** Replacing the heterogeneous attention layer (HetGAT) with homogeneous GAT/GCN/Graph Transformer variants degrades graph-constraint accuracy substantially, indicating the importance of modeling distinct room-furniture relational types during denoising.

Slots	FID ↓	KID ↓ ($\times 100$)	Coll. % ↓	Graph % ↑	Size ↑	Loc. ↑	Angle ↑
4	$7.41_{\pm 0.15}$	$0.30_{\pm 0.02}$	$13.65_{\pm 0.11}$	$89.66_{\pm 0.36}$	0.372	1.497	0.925
6 (ours)	$\mathbf{7.13}_{\pm 0.14}$	$\mathbf{0.29}_{\pm 0.01}$	$\mathbf{11.21}_{\pm 0.19}$	$\mathbf{92.34}_{\pm 0.14}$	**0.381**	**1.518**	**0.955**
8	$7.38_{\pm 0.05}$	$0.33_{\pm 0.01}$	$12.44_{\pm 0.20}$	$89.11_{\pm 0.53}$	0.337	1.512	0.903

Table 2. **Ablation on the number of LRF slots.** The number of global relation prototypes K controls the capacity of the slot dictionary. We report size, location, and angle diversity separately, following the diversity breakdown used elsewhere in the paper. A moderate number of slots provides the best balance between fidelity, structural validity, and diversity.

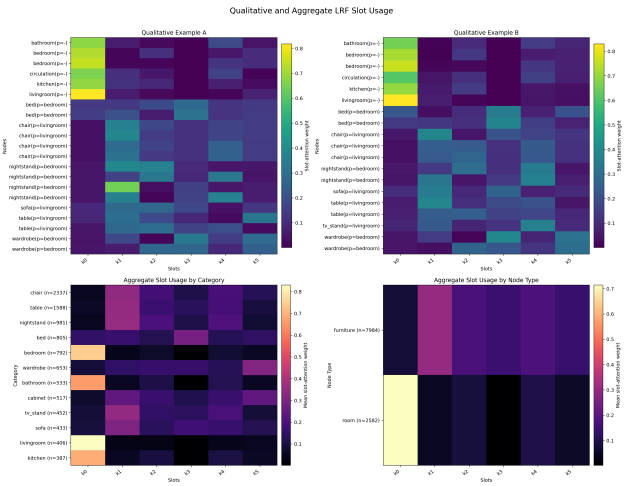

Figure 4. **Qualitative and aggregate LRF slot usage.** Top: node-by-slot heatmaps for two representative houses, with rows ordered by room nodes first and furniture nodes second. Bottom-left: mean slot usage over the full test split for the most frequent semantic categories. Bottom-right: mean slot usage grouped by node type. The learned slot dictionary exhibits a consistent room-versus-furniture separation, with room nodes concentrating on $k0$ and furniture nodes distributing mass over several secondary slots.

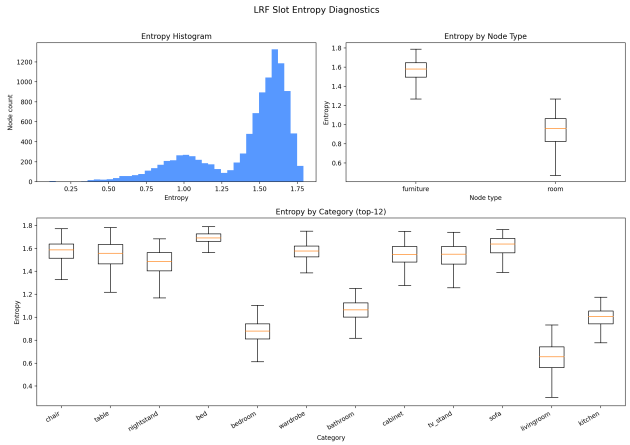

Figure 5. **Entropy diagnostics for LRF slot routing.** Left: entropy histogram over all analyzed room and furniture nodes. Top-right: entropy grouped by node type. Bottom: entropy grouped by the most frequent semantic categories. Room nodes exhibit substantially lower entropy than furniture nodes, indicating sharper and more concentrated routing, whereas furniture nodes rely on broader mixtures of relational prototypes.

D.4. Qualitative and Statistical Analysis of LRF Slots

To better understand the behavior of the Low-Rank Relational Field (LRF), we analyze the node-wise slot-attention weights induced by the trained final model. Our goal is not to assign exact symbolic meanings to individual slots, but to test whether the learned slot dictionary behaves as a structured relational vocabulary rather than a collapsed global code. For this diagnostic, we exclude the virtual house node and focus on room and furniture nodes.

Figure 4 shows that slot usage is clearly non-uniform and strongly type-dependent. In both qualitative examples, room nodes form a coherent block with sharp concentration on $k0$, whereas furniture nodes spread across several secondary slots. The aggregate summaries confirm that this pattern is not anecdotal: averaged over the full test split, room categories remain strongly aligned with $k0$, while furniture categories such as chairs, tables, nightstands, sofas, and wardrobes exhibit distinct mixtures over $k1$–$k5$. This indicates that the LRF learns multiple reusable relational

Rank	FID ↓	KID ↓ (×100)	Coll. % ↓	Graph % ↑	Size ↑	Loc. ↑	Angle ↑
16	$7.47_{\pm 0.15}$	$0.36_{\pm 0.02}$	$11.92_{\pm 0.12}$	$89.55_{\pm 0.51}$	0.328	1.517	0.909
32 (ours)	$\mathbf{7.13}_{\pm 0.14}$	$\mathbf{0.29}_{\pm 0.01}$	$\mathbf{11.21}_{\pm 0.19}$	$\mathbf{92.34}_{\pm 0.14}$	**0.381**	1.518	**0.955**
64	$7.48_{\pm 0.21}$	$0.34_{\pm 0.02}$	$12.12_{\pm 0.19}$	$88.14_{\pm 0.18}$	0.334	**1.518**	0.929

Table 3. **Ablation on the LRF rank size.** The low-rank dimension r controls the capacity of the relational field. We report size, location, and angle diversity separately, following the diversity breakdown used elsewhere in the paper. A moderate rank provides the best trade-off between fidelity, structural validity, and diversity.

Method	Time (s) ↓	Peak VRAM (MB) ↓
Two-Stage	$1.27 + 6.01 = 7.28$	$\max(113, 1017) = 1017$
Nestwork	**2.12**	**70.5**

Table 4. **Inference efficiency comparison.** Runtime and peak memory are measured under the same sampling protocol over 100 random samples.

prototypes rather than a single generic furniture code.

Entropy analysis in Figure 5 provides a complementary view of the same pattern. The global histogram is clearly non-unimodal, and the node-type boxplot shows that room nodes occupy a lower-entropy regime while furniture nodes are substantially more diffuse. At the category level, room categories such as living rooms and bedrooms remain relatively sharp, whereas frequent furniture categories exhibit higher entropy and greater variation. Together, Figures 4 and 5 indicate that the learned slot dictionary is type-dependent, systematically reused across the test set, and does not collapse to a single average relational prototype. This structured routing behavior is also consistent with the module ablation in the main manuscript: removing LRF increases collisions and degrades walkability and fidelity, suggesting that the learned slot dictionary provides useful relational bias when explicit geometric edge features are unavailable.

D.5. Inference Time and Peak VRAM

We compare the runtime and peak memory usage of Nestwork against the two-stage HouseDiffusion + DiffuScene baseline using 100 random samples on an RTX 4060 Ti. Nestwork uses 200 DDPM steps without DDIM acceleration, while the two-stage baseline uses 100 DDIM steps for each stage. Since both HouseDiffusion and DiffuScene originally use T=1000, we adopt 100-step DDIM sampling for both to keep the runtime comparison fair.

Nestwork is approximately $3.4\times$ faster and uses approximately $14\times$ less peak VRAM than the two-stage baseline. This efficiency gain follows from generating the full house in one pass rather than invoking separate room-layout and room-level furniture generators sequentially.

D.6. Diffusion Time Steps

Setup. We conduct an ablation study to evaluate how the number of diffusion denoising steps (T) affects generation quality. While the main model uses $T = 200$, we experiment with four settings: $T = 50$, 100, 200, and 500. All variants use identical architectures, losses, and training schedules, and are evaluated under full semantic graph conditioning. We report FID/KID for visual realism, graph-constraint accuracy for structural validity, and collision rate for physical feasibility, averaged over three random seeds in Table 5.

Discussion. The results show a clear relationship between the number of diffusion steps and layout quality. Very small schedules (e.g., T=50) under-denoise the latent graph, producing the worst FID and KID as well as the highest collision rate. Increasing to T=100 yields substantial improvements but still falls noticeably short of the main setting. The T=200 model achieves the best overall performance across all core metrics–*lowest FID, lowest KID, and lowest collision rate*–while maintaining strong graph-constraint accuracy. Although T=50 attains the highest graph accuracy, the difference compared to T=200 is small (92.69% vs. 92.34%) and does not outweigh the large losses in visual fidelity and physical plausibility. Finally, increasing to

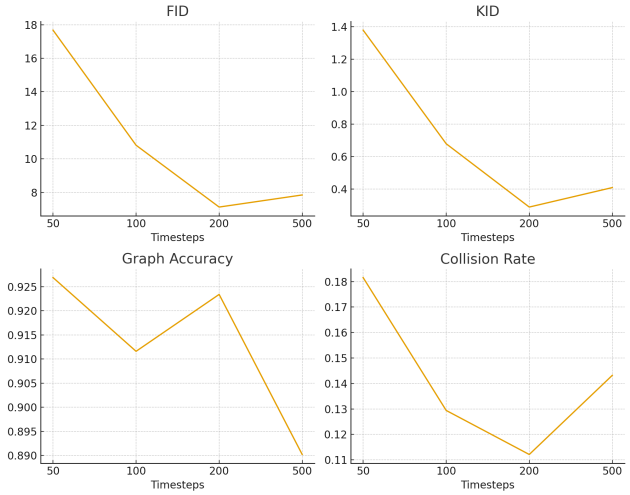

Figure 6. Diffusion Timestep Ablation Study

T=500 does not bring further benefits: FID slightly worsens relative to T=200, collision rate increases, and graph accuracy drops significantly. These observations indicate that T=200 provides the best trade-off between realism, stability, and structural correctness, justifying its use as our default configuration.

D.7. Masking Ratio Ablation

Setup. We study how the node-label masking ratio used during diffusion training affects downstream generation. We keep all components, data splits, and optimization hyperparameters fixed and vary only the masking policy: (i) a fixed 25% masking ratio, (ii) a fixed 50% ratio (used in the main manuscript), (iii) a fixed 75% ratio, and (iv) a uniform schedule that samples a masking ratio from $[0, 1]$ for each training batch. For each policy, we train a single diffusion model and evaluate it under the four conditioning modes introduced in the main experiments in Section 4.7: *FullGraph*, *RoomGraph*, *Topology*, and *RandMask* (50% random masking at test time). We report FID/KID, graph-constraint accuracy, collision rate, and diversity metrics (size, location, angle).

Figure 7. Masking Ratio Ablation Study

Discussion. Across all conditioning modes, a moderate fixed masking ratio provides the best balance of fidelity and robustness. Uniform masking performs the worst overall, especially in collision and graph accuracy, suggesting that exposing the model to highly inconsistent supervision harms stability. Low masking (25%) slightly improves graph accuracy but degrades FID/KID, while high masking (75%) over-regularizes the denoiser and suppresses semantic detail. The 50% setting consistently achieves the strongest FID/KID and competitive collision rates, indicating that a mid-level masking ratio supplies enough semantic context while still training the model to recover missing structure–aligning well with the intended use cases of partially specified conditioning graphs.

D.8. Latent Dimension Size

In Section 4.5 (Autoencoder Backbone Study) we compared different GNN backbones under a fixed latent size (64 for *SLN-Box* and 128 for *Nestwork* and *TripletGCN*). Here we study how latent capacity alone affects generation quality. We sweep the latent dimension in our autoencoder and ask whether larger latents improve realism and constraint satisfaction enough to justify their cost.

Setup. We fix the Nestwork backbone (HetGAT encoder and decoder), the latent diffusion prior, and the denoising HetGAT backbone. We vary the latent dimension across four values: 32, 64, 128 (ours), and 256. All other configurations remain unchanged: (1) full semantic graph conditioning (G_{sem}); (2) training schedule, optimizer, and dataset split as in Section 4; and (3) evaluation under FID, KID, graph-constraint satisfaction (%), and furniture collision rate (%).

Discussion. Small latents (32) lack capacity and hurt both realism and constraint satisfaction. Our 128-dimensional latent offers a better balance: slightly higher FID than 64, but noticeably lower collision, higher diversity, and the best graph accuracy. Pushing capacity to 256 is clearly detrimental, leading to severe degradation across all metrics and suggesting overfitting and optimization instability. We therefore adopt 128 as the default latent size, as it provides sufficient expressiveness without sacrificing robustness.

Timesteps	**FID** ↓	**KID** ↓ (×100)	**Coll.** %↓	**Graph** %↑
50	$17.68_{\pm 0.23}$	$1.38_{\pm 0.04}$	$18.16_{\pm 0.25}$	$\mathbf{92.69}_{\pm 0.50}$
100	$10.81_{\pm 0.43}$	$0.68_{\pm 0.04}$	$12.94_{\pm 0.26}$	$91.16_{\pm 0.50}$
200 (ours)	$\mathbf{7.13}_{\pm 0.14}$	$\mathbf{0.29}_{\pm 0.01}$	$\mathbf{11.21}_{\pm 0.19}$	$92.34_{\pm 0.14}$
500	$7.85_{\pm 0.16}$	$0.41_{\pm 0.02}$	$14.32_{\pm 0.10}$	$89.02_{\pm 0.29}$

Table 5. **Effect of diffusion time steps.** The main model ($T = 200$) achieves the best overall balance between fidelity, stability, and structural accuracy.

Masking	FID↓	KID↓(×100)	Coll. %↓	Graph %↑	Size↑	Loc.↑	Angle↑
25%	$8.48_{\pm 0.27}$	$0.49_{\pm 0.05}$	$11.53_{\pm 0.23}$	$\mathbf{93.72}_{\pm 0.28}$	0.35	1.52	**0.96**
75%	$9.64_{\pm 0.03}$	$0.61_{\pm 0.01}$	$\mathbf{10.31}_{\pm 0.16}$	$93.64_{\pm 0.47}$	0.36	1.53	0.95
Uniform	$8.53_{\pm 0.26}$	$0.45_{\pm 0.03}$	$13.35_{\pm 5.00}$	$91.95_{\pm 0.49}$	**0.40**	**1.54**	0.93
50% (ours)	$\mathbf{7.13}_{\pm 0.14}$	$\mathbf{0.29}_{\pm 0.01}$	$11.21_{\pm 0.19}$	$92.34_{\pm 0.14}$	0.38	1.52	0.95

(a) FullGraph conditioning

Masking	FID↓	KID↓(×100)	Coll. %↓	Graph %↑	Size↑	Loc.↑	Angle↑
25%	$11.45_{\pm 0.07}$	$0.75_{\pm 0.02}$	$\mathbf{10.09}_{\pm 0.15}$	$93.07_{\pm 0.19}$	0.54	**1.59**	0.92
75%	$9.06_{\pm 0.25}$	$0.55_{\pm 0.04}$	$10.72_{\pm 0.21}$	$\mathbf{93.17}_{\pm 0.04}$	0.50	1.55	**0.96**
Uniform	$9.18_{\pm 0.09}$	$0.46_{\pm 0.01}$	$14.68_{\pm 0.24}$	$91.07_{\pm 0.26}$	**0.57**	1.55	0.94
50% (ours)	$\mathbf{7.26}_{\pm 0.43}$	$\mathbf{0.26}_{\pm 0.04}$	$10.91_{\pm 0.21}$	$91.91_{\pm 1.01}$	0.55	1.55	0.95

(b) RoomGraph conditioning

Masking	FID↓	KID↓(×100)	Coll. %↓	Graph %↑	Size↑	Loc.↑	Angle↑
25%	$12.72_{\pm 0.19}$	$0.84_{\pm 0.03}$	$\mathbf{10.08}_{\pm 0.13}$	$\mathbf{92.45}_{\pm 0.13}$	0.57	**1.59**	0.93
75%	$9.76_{\pm 0.30}$	$0.61_{\pm 0.03}$	$10.69_{\pm 0.14}$	$92.40_{\pm 0.12}$	0.50	1.54	**0.96**
Uniform	$9.68_{\pm 0.13}$	$0.51_{\pm 0.01}$	$14.87_{\pm 0.08}$	$90.89_{\pm 0.29}$	**0.58**	1.56	0.94
50% (ours)	$\mathbf{7.75}_{\pm 0.15}$	$\mathbf{0.29}_{\pm 0.01}$	$10.60_{\pm 0.29}$	$92.00_{\pm 0.65}$	0.56	1.55	0.95

(c) Topology conditioning

Masking	FID↓	KID↓(×100)	Coll. %↓	Graph %↑	Size↑	Loc.↑	Angle↑
25%	$9.07_{\pm 0.10}$	$0.54_{\pm 0.01}$	$11.50_{\pm 0.14}$	$\mathbf{93.41}_{\pm 0.23}$	0.51	**1.56**	**0.98**
75%	$9.63_{\pm 0.27}$	$0.61_{\pm 0.04}$	$\mathbf{10.38}_{\pm 0.11}$	$93.07_{\pm 0.18}$	0.47	1.54	0.96
Uniform	$9.04_{\pm 0.08}$	$0.47_{\pm 0.01}$	$14.01_{\pm 0.24}$	$91.71_{\pm 0.05}$	**0.54**	**1.56**	0.94
50% (ours)	$\mathbf{6.96}_{\pm 0.03}$	$\mathbf{0.27}_{\pm 0.02}$	$11.16_{\pm 0.05}$	$92.59_{\pm 0.11}$	0.51	1.54	0.96

(d) RandMask conditioning

Table 6. **Masking-ratio ablation across conditioning modes.** Each subtable reports the effect of training-time masking ratio on FID/KID, structural constraint satisfaction, collision rate, and diversity under a specific conditioning mode.

Latent Dim	FID↓	KID↓(×100)	Coll. %↓	Graph %↑	Size↑	Loc.↑	Angle↑
32	$8.94_{\pm 0.18}$	$0.52_{\pm 0.03}$	$13.08_{\pm 0.19}$	$90.24_{\pm 0.21}$	0.33	1.46	**1.05**
64	$\mathbf{6.92}_{\pm 0.17}$	$0.29_{\pm 0.03}$	$14.53_{\pm 0.29}$	$92.15_{\pm 0.13}$	0.35	1.44	0.89
128 (ours)	$7.13_{\pm 0.14}$	$\mathbf{0.29}_{\pm 0.01}$	$\mathbf{11.21}_{\pm 0.19}$	$\mathbf{92.34}_{\pm 0.14}$	**0.38**	**1.52**	0.95
256	$34.62_{\pm 0.55}$	$2.66_{\pm 0.08}$	$53.22_{\pm 0.67}$	$79.22_{\pm 1.07}$	0.18	0.86	0.53

Table 7. **Latent dimension ablation.** Increasing the latent from 32 to 64 and 128 improves realism and constraints, but a very large latent (256) collapses performance, suggesting over-parameterization and unstable optimization. We adopt 128 as the default latent size.

E. Limitations

Our current system targets early-stage ideation; it does not yet guarantee fully code-compliant layouts. First, several failure modes persist–occasional furniture interpenetration and tight clearances (Fig. 8). These issues stem from design choices that favor scalability and generality: the graph schema omits wall thickness, door positions, and explicit circulation intent; mesh retrieval can mis-fit predicted boxes; and diffusion optimizes likelihood rather than hard feasibility.

Second, functional validation remains outside the model loop. While Nestwork already supports mask-based conditioning and LLM prompts, true production use will require richer signals–doors, egress widths, accessibility rules–and post-hoc or learned critics that score clearance, daylight, or HVAC zones. Integrating such validators, along with interactive fixed-subgraph inpainting or co-optimized box-and-mesh generation, is promising future work that complements–rather than replaces–our core contribution of heterogeneous graph auto-encoding with graph-based diffusion.

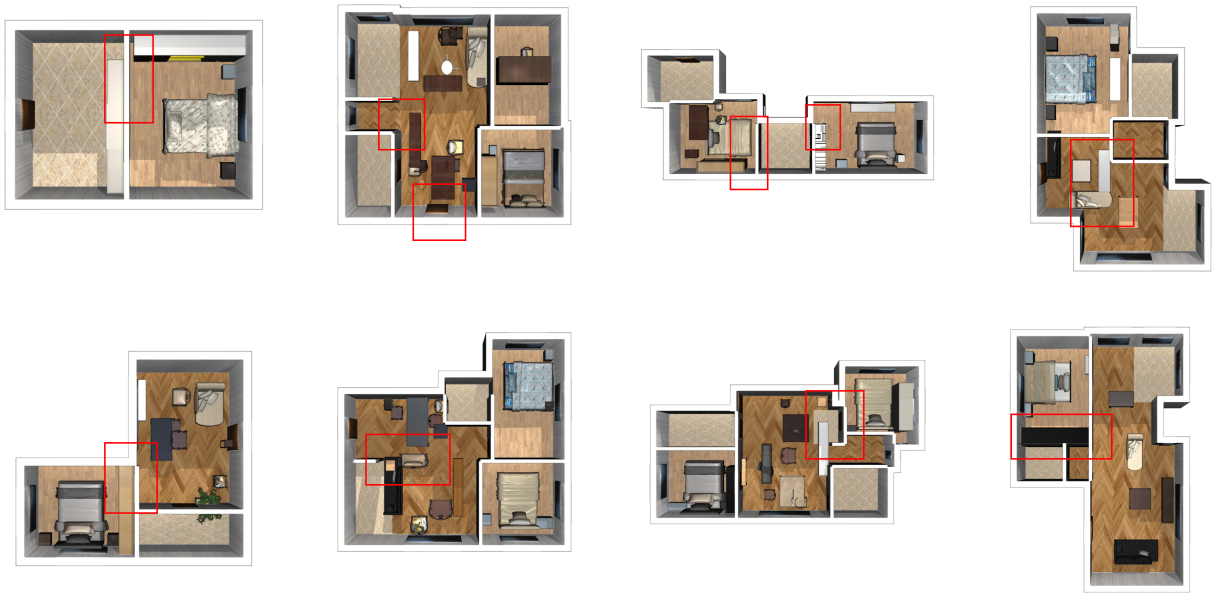

Figure 8. **Limitations.** Examples of failure cases produced by Nestwork. Red boxes highlight circulation congestion and tight clearances, where furniture placement restricts access or violates expected egress space.

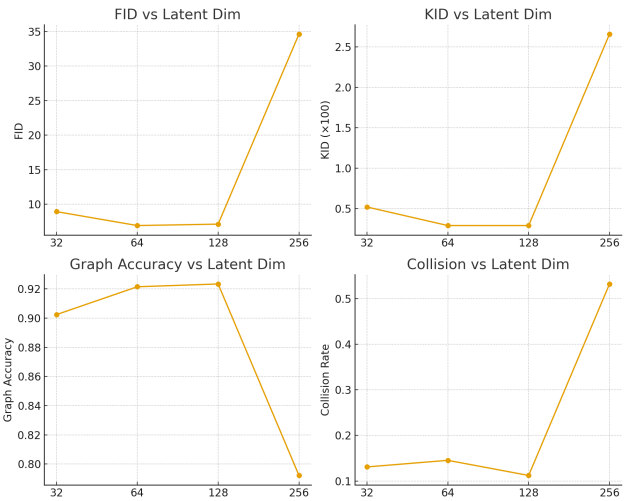

Figure 9. Effect of latent dimension size.

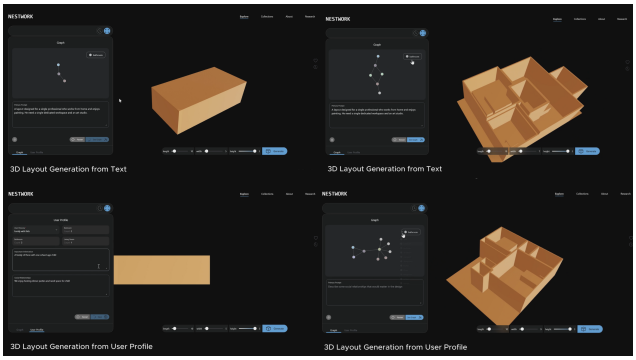

Figure 10. Web-interface prototype. A user enters either (i) a free-form description or (ii) a structured profile. The LLM converts the text into a *masked* semantic graph—*rooms* ***and*** *their furniture counts* may be fully specified, partially specified, or left as `[MASK]`—and, when enabled, a matching structured prompt for the auxiliary text branch. The graph is fed to Nestwork for layout generation; the current demo shows bounding boxes only.

F. Natural-Language Interface

F.1. LLM Prompting for Masked Graphs

Our diffusion model inpaints from a masked semantic graph; when the auxiliary text branch is enabled, the LLM also emits a matching structured prompt. The graph itself is represented by four lists: **Room List**, **Furniture Dict**, **Adjacency List**, and **Separation List**. A label may be a concrete type (*bedroom*, *sofa*) or the token `[MASK]`.

Base prompt. For free descriptions (e.g., "a compact apartment with a study corner") the LLM (i) infers occupants, (ii) decides required rooms, (iii) lists typical furniture for each room, replacing uncertain items with `[MASK]`, and (iv) proposes adjacency/separation pairs.

Profile prompt. Takes a JSON profile containing fields such as `bedroomNum`, `bathroomNum`, and user-preferred furniture. Missing or flexible entries are emit-

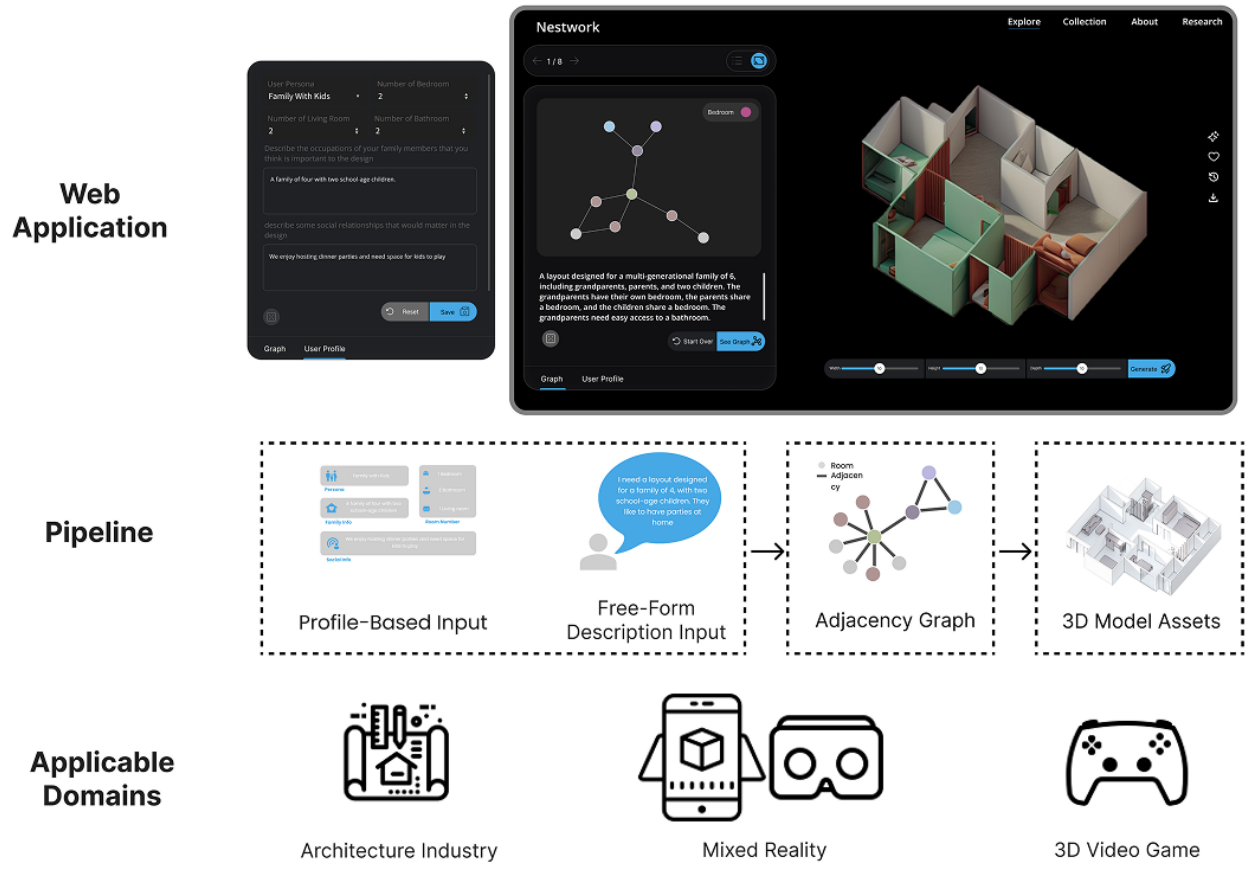

Figure 11. Web application supporting multiple domains with our proposed pipeline.

Field	Example
Room List	`"0":"bedroom","1":"livingroom","2":"kitchen","3":"[MASK]"`
Furniture Dict	`"0":"bed":1,"nightstand":2, "1":"sofa":1,"desk":"[MASK]",` `"2":"table":1,"chair":4, "3":"[MASK]":"[MASK]"`
Adjacency List	`[["1","2"],["0","1"]]`
Separation List	`[["0","2"]]`

Table 8. LLM output schema for masked-graph conditioning. Labels may be concrete types or `[MASK]`.

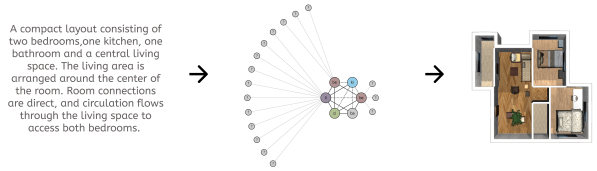

Figure 12. A natural-language prompt is parsed into a masked room + furniture graph, which Nestwork converts into a fully furnished 3D layout.

ted as `[MASK]`, so the graph can range from fully specified to largely unspecified.These templates guarantee a smooth hand-off from language to masked-graph conditioning, enabling fine-to-coarse specification without modifying the generation pipeline.

References

[1] Aditya Chattopadhyay, Xi Zhang, David Paul Wipf, Himanshu Arora, and René Vidal. Learning graph variational autoencoders with constraints and structured priors for conditional indoor 3d scene generation. *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 785–794, 2023.

[2] Helisa Dhamo, Fabian Manhardt, Nassir Navab, and Federico Tombari. Graph-to-3d: End-to-end generation and manipulation of 3d scenes using scene graphs. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[3] Harold W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 52, 1955.

[4] Andrew Luo, Zhoutong Zhang, Jiajun Wu, and Joshua B. Tenenbaum. End-to-end optimization of scene layout. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3753–3762, 2020.

[5] Despoina Paschalidou, Amlan Kar, Maria Shugrina, Karsten Kreis, Andreas Geiger, and Sanja Fidler. Atiss: Autoregressive transformers for indoor scene synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[6] Mohammad Amin Shabani, Sepidehsadat Hosseini, and Yasutaka Furukawa. Housediffusion: Vector floorplan generation via a diffusion model with discrete and continuous denoising. *arXiv preprint arXiv:2211.13287*, 2022.

[7] Chong Su, Yingbin Fu, Zheyuan Hu, Jing Yang, Param Hanji, Shaojun Wang, Xuan Zhao, Cengiz Öztireli, and Fangcheng Zhong. Chord: Generation of collision-free, house-scale, and organized digital twins for 3d indoor scenes with controllable floor plans and optimal layouts. *arXiv preprint arXiv:2503.11958*, 2025.

[8] Jiapeng Tang, Yinyu Nie, Lev Markhasin, Angela Dai, Justus Thies, and Matthias Nießner. Diffuscene: Denoising diffusion models for generative indoor scene synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[9] Guangyao Zhai, Evin Pınar Örnek, Shun-Cheng Wu, Yan Di, Federico Tombari, Nassir Navab, and Benjamin Busam. Commonscenes: Generating commonsense 3d indoor scenes with scene graph diffusion. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.