DeBaRA: Denoising-Based 3D Room Arrangement Generation

Léopold Maillard[1,2] **Nicolas Sereyjol-Garros**[*] **Tom Durand**[2] **Maks Ovsjanikov**[1]

[1]LIX, École Polytechnique, IP Paris [2]Dassault Systèmes

{maillard,maks}@lix.polytechnique.fr {firsname.lastname}@3ds.com

Abstract

Generating realistic and diverse layouts of furnished indoor 3D scenes unlocks multiple interactive applications impacting a wide range of industries. The inherent complexity of object *interactions*, the limited amount of available data and the requirement to fulfill spatial constraints all make generative modeling for 3D scene synthesis and arrangement challenging. Current methods address these challenges autoregressively or by using off-the-shelf diffusion objectives by simultaneously predicting all attributes without 3D reasoning considerations. In this paper, we introduce DeBaRA, a score-based model specifically tailored for precise, controllable and flexible arrangement generation in a bounded environment. We argue that the most critical component of a scene synthesis system is to accurately establish the *size* and *position* of various objects within a restricted area. Based on this insight, we propose a lightweight conditional score-based model designed with 3D spatial awareness at its core. We demonstrate that by focusing on spatial attributes of objects, a single trained DeBaRA model can be leveraged at test time to perform several downstream applications such as scene synthesis, completion and re-arrangement. Further, we introduce a novel *Self Score Evaluation* procedure so it can be optimally employed alongside external LLM models. We evaluate our approach through extensive experiments and demonstrate significant improvement upon state-of-the-art approaches in a range of scenarios.

1 Introduction

Systems capable of generating realistic environments comprising multiple interacting objects would impact several industries including video games, robotics, augmented and virtual reality (AR/VR) and computer-aided interior design. As a result and in tandem with the growing availability of synthetic datasets of indoor layouts [10, 44, 42, 63, 7], which can be populated with high-quality 3D assets [11, 63, 1], data-driven approaches for automatically generating and arranging 3D scenes have been actively investigated by the computer vision community. Notably, the ongoing success of deep generative models for controllable content creation in the text and image domains has recently been extended to scene synthesis, allowing users to craft realistic indoor environments from a set of multimodal constraints [38, 37, 55, 54, 27, 35].

Challenges associated with 3D indoor scene generation are numerous as the intricate nature of multi-object interactions is difficult to capture and model precisely. Items should be placed, potentially resized and oriented relative to one another, in a way that is both plausible and aligned with subjective and context-dependent priors such as style, as well as ergonomic and functional preferences. Additionally, objects should fit within a bounded, restricted area, and a subtle mismatch can break the perceived validity of the synthesized environment (e.g., overlapping, floating or out-of-bounds

[*]Work done during internship at Dassault Systèmes.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

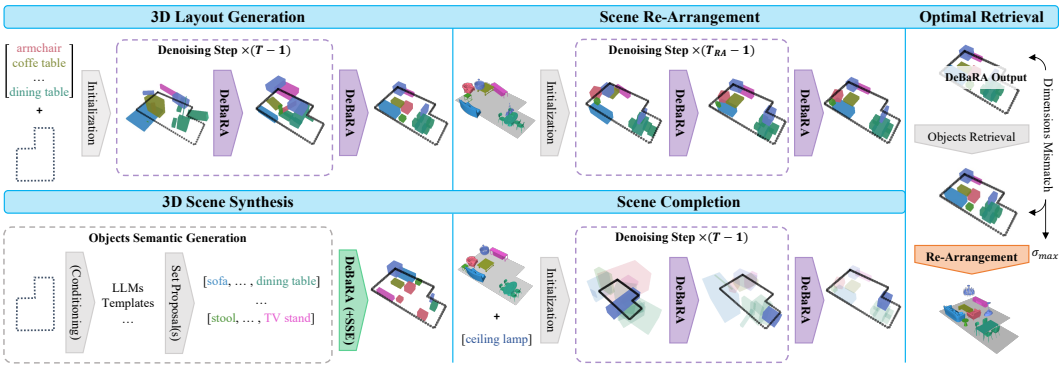


Figure 1: Application scenarios overview. Besides generating diverse and realistic 3D indoor layouts, a single trained DeBaRA model can be employed to execute several related tasks by tweaking the initial sampling noise level σ_{max} and/or performing object or attribute-level layout inpainting. Our novel SSE procedure enables 3D Scene Synthesis capabilities by efficiently selecting conditioning semantics from external sources using density estimates provided by the pretrained model.

objects, inaccessible areas). Finally, the limited availability of high-quality data [10, 42] requires learning-based approaches to make careful design choices and trade-offs.

Early data-driven approaches often rely on intermediate hand-crafted representations [43, 54, 36, 62] that are closely related to the considered dataset, which introduces significant biases. Concurrently, popular methods have been adopting autoregressive architectures that treat scene synthesis as a set generation task [55, 38, 27, 21, 37] by sequentially adding individual objects. More recently, score-based generative models (also known as *denoising diffusion models*) have shown promising capabilities in various 3D scene understanding applications [18, 58] including controllable scene synthesis [51, 62, 60] and re-arrangement [57]. In contrast to previous methods, denoising-based approaches enable a stable and scalable training phase and can output all scene attributes simultaneously. The iterative sampling framework brings an improved consideration for the conditioning information and an attractive balance between generation quality and variety. However, current methods leveraging score-based generative models try to model all attributes (both categorical and spatial) within a single framework, which, as we demonstrate below, is less data-efficient and leads to suboptimal solutions.

In this context, our work aims to establish principled and robust capabilities for generating accurate and diverse 3D layouts. Specifically, our key contributions are threefold:

1. We propose a score-based conditional objective and architecture designed to effectively learn spatial attributes of interacting 3D objects in a constrained indoor environment. In contrast to previous approaches [51, 38], we disentangle the design space and reduce the model's prediction to a minimal representation consisting solely of oriented 3D bounding boxes, taking as conditioning input the room's floor plan and list of object semantic categories.
2. We propose a set of approaches which allows a model trained following our method to be flexibly employed at test time to perform several user-driven tasks enabling object or attribute-level control. In particular, we demonstrate strong capabilities on controllable scenarios such as scene re-arrangement or room completion, from a single trained network.
3. Finally, we introduce a novel *Self Score Evaluation* (SSE) procedure, which enables 3D scene synthesis by selecting the set of inputs provided by external sources, such as a LLM, that lead to the most realistic layouts.

We exhibit our model's capabilities across a wide range of experimental scenarios and report state-of-the art 3D layout generation and scene synthesis performance.

2 Related Work

Score-based Generative Models By smoothly perturbing training examples with noise, Diffusion Models map a complex data distribution to a known Gaussian prior from which they sample back via

iterative denoising using a neural network trained over multiple noise levels. This family of generative models has been motivated by several theoretical foundations over the past years: DDPMs [16, 33] parameterize the diffusion process as a discrete-time Markov chain, as opposed to *continuous-time* approaches [49, 48]. The seminal EDM [19, 20] training and sampling settings later unified previous methods into an improved ideal framework defined by a set of interpretable parameters. Originally motivated by image generation, diffusion models have demonstrated impressive capabilities on various conditional tasks such as text-to-image synthesis [34, 45], image-to-image generation from various 2D input modalities [45, 64, 56], text-to-3D asset creation [40, 23] or environment-aware human motion synthesis [18, 24]. Relevant to our work, diffusion models have been applied to the generation of point clouds [31, 52] and other geometric representations involving 3D coordinates [59].

Lifting Pretrained Diffusion Models Knowledge of trained diffusion models can be leveraged in various settings including content inpainting[30, 18], score distillation [40], exact likelihood computation [49, 19] or teacher-student distillation [47, 32]. More relevant to our work, image-domain diffusion priors have demonstrated compelling performance in discriminative tasks including zero-shot image classification [22, 6, 5] and segmentation [4]. More precisely, Diffusion Classifiers assign a label, from a finite set of possible classes $\{\boldsymbol{c}_i\}_{i=1}^{N}$ to an observed sample $\boldsymbol{x}_0$ by computing class-conditional density estimates from a pretrained diffusion model under the assumption of a uniform prior $p(\boldsymbol{c}_i) = 1/N$. In practice, this is done by, for each class, iteratively adding noise to the observed sample $\boldsymbol{x}_0$ and computing a Monte Carlo estimate of the expected reconstruction loss using the class-conditioned model.

Controllable 3D Scene Synthesis Synthesizing indoor 3D layouts from a partial set of information or constraints has come in various settings depending on provided vs. predicted entities and enabled control granularity. A prolific line of research has been adopting intermediate 3D scene representations such as graphs [25, 43, 54, 36, 62, 13, 26], furniture matrices [65] or multi-view images [35]. Autoregressive furnishing approaches [55, 38] have been supplemented by object attribute-level conditioning [37, 27] and additional ergonomic constraints [21]. However, their *one object at a time* strategy does not comprehensively capture complex relationships between all the interacting elements and is known to easily fall into local minima in which new items fail to be accurately inserted to the current configuration. Lately, methods have unfolded LLMs double-edged capabilities in this area [9, 61] as they excel at generating sensible furniture descriptions while struggling in accurately arranging them in the 3D space, which [12] addresses by introducing a costly refinement stage. In the light of that, LLMs appear to be ideal candidates to supplement a specialized 3D layout generation model.

Denoising Indoor Scenes Previous methods have explored diffusion-based approaches in the context of 3D scene synthesis. Pioneering their usage, LEGO-Net [57] performs scene re-arrangement (i.e., recovering a *clean* object layout from a *noisy* one) in the 2D space using a transformer backbone that is not noise-conditioned, which we argue is the root cause of its main limitations. PhyScene[60] augment diffusion-based 3D scene synthesis with additional physic-based guidance to enable practical embodied agent applications. Most relevant to our work, DiffuScene [51] achieves 3D scene synthesis by fitting a DDPM [16] on stacked 3D object features, resulting in a high-dimensional composite distribution that is hard to learn and interpret. It does not enforce spatial configurations over other predicted features. More importantly, its generative process is not conditioned on the room's floor plan (i.e., bounds) that constrains objects to be placed within a restricted area.

3 Method

3.1 3D Scene Representation

Our method is based on encoding the state of a 3D indoor scene $\mathcal{S}$ that is defined by a floor plan (i.e., bounds) $\mathcal{F}$ and an unordered set of N objects $\mathcal{O} = \{o_1, \dots, o_N\}$, each being modeled by its typed 3D bounding box $o_i = \{\boldsymbol{x}_i, \boldsymbol{c}_i\}$ where $\boldsymbol{c}_i \in \{0, 1\}^k$ is the one-hot encoding of the semantic category among k classes and $\boldsymbol{x}_i = (\boldsymbol{p}_i, \boldsymbol{r}_i, \boldsymbol{d}_i) \in \mathbb{R}^8$ comprises 3D spatial attributes. More specifically, $\boldsymbol{p}_i \in \mathbb{R}^3$ denotes the object's center coordinate position, $\boldsymbol{r}_i = (\cos\theta_i, \sin\theta_i) \in \mathbb{R}^2$ is a continuous encoding of the rotation of angle θ_i around the scene's vertical axis [66] and $\boldsymbol{d}_i \in \mathbb{R}^3$ is the dimension.

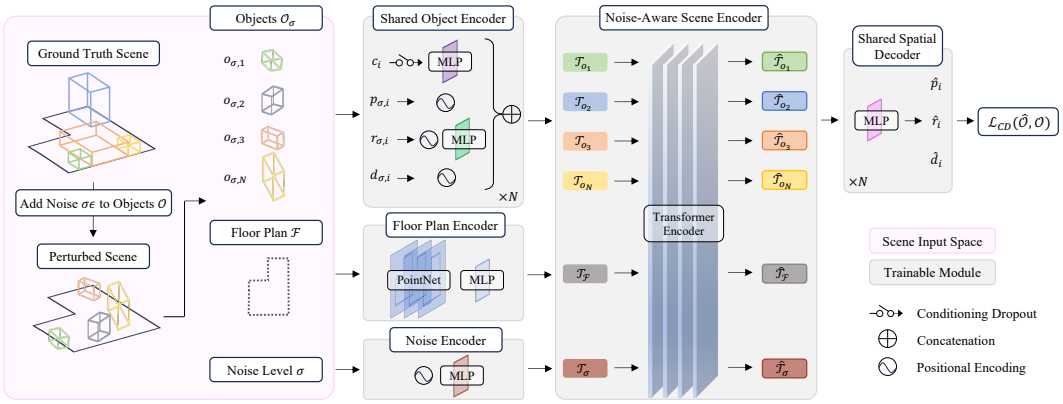


Figure 2: DeBaRA architecture and training overview. At each iteration, 3D bounding boxes parameters $(\boldsymbol{p}, \boldsymbol{r}, \boldsymbol{d})$ of indoor scene's objects $\mathcal{O}$ are perturbed with Gaussian noise $\sigma\boldsymbol{\epsilon}$. The floor plan $\mathcal{F}$, noise level σ and resulting objects $\mathcal{O}_\sigma$ are processed by respective encoders to form an unordered set of representations $\mathcal{T}$ fed as input to a transformer encoder. Novel object embeddings $\hat{\mathcal{T}}_o$ are finally decoded back to their predicted *clean* spatial configuration $(\hat{\boldsymbol{p}}, \hat{\boldsymbol{r}}, \hat{\boldsymbol{d}})$. Trainable modules are optimized by minimizing a semantic-aware Chamfer loss. Input object categories $\boldsymbol{c}$ are randomly dropped to model both the class-conditional and unconditional 3D layout distributions.

3.2 Diffusion Framework and Architecture

We describe in this section our score-based layout generation framework, relevant design choices and network architecture, that are summarized in Figure 2. Remarkably, unlike previous approaches [51, 38, 37] that output a range of attributes lying in different spaces, we focus on accurately modeling 3D spatial layouts of bounded indoor scenes from a set of input object categories.

Learning 3D spatial configurations from object semantics We adopt a diffusion-based approach to yield a conditional generation model that outputs 3D object spatial features $\{\boldsymbol{x}_i\}_{i=1}^N$ from an input floor plan and set of semantic categories $\boldsymbol{y} = (\mathcal{F}, \boldsymbol{c})$ with $\boldsymbol{c} = \{\boldsymbol{c}_i\}_{i=1}^N$. During training, 3D spatial attributes are perturbed with Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ at various noise levels (i.e., magnitudes) σ. A trainable noise-conditioned denoiser model $D_\theta(\boldsymbol{x}_\sigma; \boldsymbol{y}, \sigma)$ maps *noisy* spatial attributes $\boldsymbol{x}_\sigma = \boldsymbol{x} + \sigma\boldsymbol{\epsilon}$ to their *clean* counterparts $\hat{\boldsymbol{x}} \approx \boldsymbol{x} \in \mathbb{R}^{N\times 8}$.

We notice that each object spatial attribute has an individual real-world interpretation (e.g, $\boldsymbol{p}$ and $\boldsymbol{d}$ can be expressed in meters, $\boldsymbol{r}$ in degrees). To preserve their measurable nature at intermediate perturbed configurations $\boldsymbol{x}_\sigma$, we want our diffusion parameterization to support a continuous range of noise levels, correlated to the scale of the input signal. This will be particularly convenient at test time (see Section 3.5). To guarantee both properties, we adapt the score-based EDM [19] framework. In our context, this formulation is more natural than the DDPM framework employed by previous work [51]. The latter is based on *discretizing* noise levels and does not offer a straightforward interpretability of the scene's state at arbitrary timesteps. Our parameterization is further detailed in Appendix A.

Estimating the unconditional layout density Inspired by *classifier-free guidance* [17] in the image domain, we model both the class-conditional density $p_\theta(\boldsymbol{x}|\mathcal{F}, \boldsymbol{c})$ and the unconditional density $p_\theta(\boldsymbol{x}|\mathcal{F}, \emptyset)$ by a single network of parameters θ. At each training iteration, we perform *conditioning dropout* on the set of semantic categories, by setting $\boldsymbol{c} = \emptyset$ with probability p_{drop} else $\{\boldsymbol{c}_i\}_{i=1}^N$. We found that this mechanism helps to reduce overfitting of the training layouts $p_{\text{data}}(\boldsymbol{x})$ and enables novel capabilities that we introduce in Section 3.4.

Denoiser Network Architecture Our lightweight architecture is inspired by previous work [57] to which we make key changes. Similar to [38] and [51], we use a shared object encoder in order to obtain per-object token $\mathcal{T}_{o_i}$ as a concatenation of the object o_i attributes embedded by sinusoidal positional encoding and linear layers. Following [57], we uniformly sample P points on the edges of the floor plan and feed them into a PointNet [41] model, resulting in a floor token $\mathcal{T}_\mathcal{F}$. This choice of feature extractor backbone is natural as it allows to maintain all the input scene's spatial features in a

common 3D space. Importantly, a noise token $\mathcal{T}_\sigma$ is computed from the current noise level σ, making our architecture *noise-aware*, i.e., able to denoise layouts $\boldsymbol{x}_\sigma$ at any perturbation magnitude.

All the previously encoded tokens form a sequence $\mathcal{T} = \{\mathcal{T}_\mathcal{F}, \mathcal{T}_\sigma, \mathcal{T}_{o_i}, \dots, \mathcal{T}_{o_N}\}$ from which a global scene encoder T_θ computes rich representations $\hat{\mathcal{T}}$. We design the method without any token ordering and use padding mask for scene with fewer objects than the transformer capabilities. A final shared decoder MLP takes as input object tokens $\{\hat{\mathcal{T}}_{o_i}\}_{i=1}^N$ and returns denoised spatial attribute values $\hat{\boldsymbol{x}} = \{(\hat{\boldsymbol{p}}_i, \hat{\boldsymbol{r}}_i, \hat{\boldsymbol{d}}_i)\}_{i=1}^N$. We provide implementation details on the denoiser in Appendix B.1.

3.3 3D Spatial Objective

Our noise-conditioned model D_θ is optimized towards a novel semantic-aware Chamfer Distance objective that does not penalize permutation of 3D bounding boxes sharing the same semantic category between the predicted scene objects layout $\hat{\mathcal{O}}$ and the ground truth one $\mathcal{O}$:

$$\mathcal{L}_{CD}(\hat{\mathcal{O}}, \mathcal{O}) = \frac{1}{2N}\left(\sum_{\hat{o}\in\hat{\mathcal{O}}} \min_{o\in\mathcal{O}} l(\hat{o}, o) + \sum_{o\in\mathcal{O}} \min_{\hat{o}\in\hat{\mathcal{O}}} l(\hat{o}, o)\right), \tag{1}$$

$$\text{where} \quad l(\hat{o}, o) = \|\hat{\boldsymbol{x}} - \boldsymbol{x}\|_2^2 + \kappa\big(1 - \delta_{\boldsymbol{c}}(\hat{o}, o)\big). \tag{2}$$

Here, κ is a large value so that a significant penalty is applied to objects that do not share the same semantic category $\boldsymbol{c}$, preventing them to be returned by the $\min$ operator.

We can finally rewrite the usual score-based training objective [49, 19] as:

$$\mathbb{E}_{p_{\text{data}}(\boldsymbol{x}),\boldsymbol{\epsilon},\sigma}\big[\lambda(\sigma)\mathcal{L}_{CD}(D_\theta(\boldsymbol{x} + \sigma\boldsymbol{\epsilon}; \boldsymbol{y}, \sigma), \boldsymbol{x})\big] \tag{3}$$

where $\lambda(\sigma)$ is a noise-dependent loss weighting function.

3.4 Self Score Evaluation

While specifying complete conditioning information such as the set of object semantics $\boldsymbol{c}$ could be tedious, it can be *provided* by either a LLM or a separately trained sequence generation model. However, using independent models is inherently suboptimal since it does not guarantee that the generated conditioning input will be aligned with the score model knowledge. As a result, we propose a novel method to select conditioning inputs that are attuned with the model's capabilities.

More specifically, we evaluate a finite set of C object semantic categories *candidates*, where each candidate is associated to a 3D spatial layout sampled from the learned conditional density, i.e.,

$$\text{candidates} = \left\{\big(\boldsymbol{c}_j, \boldsymbol{x}_j \sim p_\theta\big(\boldsymbol{x}|\mathcal{F}, \boldsymbol{c}_j\big)\big)\right\}_{j=1}^C \tag{4}$$

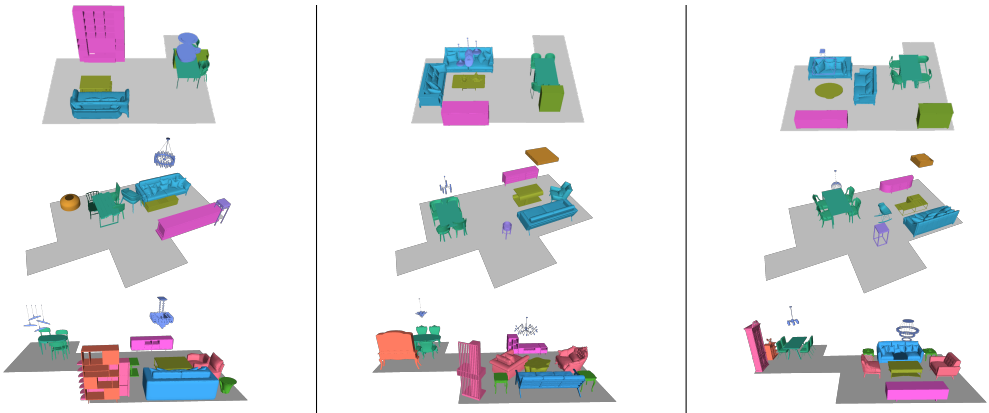
Then, the optimal conditioning candidate $\boldsymbol{c}^*$ is derived from a density estimate of its corresponding 3D spatial layout $\boldsymbol{x}^*$ provided by the unconditional network:

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}_i} \mathbb{E}_{\boldsymbol{\epsilon},\sigma}\big[\mathcal{L}_{CD}\{D_\theta(\boldsymbol{x}_i + \sigma\boldsymbol{\epsilon}\,;\, \mathcal{F}, \emptyset, \sigma), \boldsymbol{x}_i\}\big] \tag{5}$$

Algorithm 1 Self Score Evaluation

Require: a diffusion prior D_θ trained with conditioning dropout and by optimizing $\mathcal{L}_{CD}$
Input: conditioning candidates $\{\boldsymbol{c}_j\}_{j=1}^C$, number of score evaluation trials T_{sse}

sample $\boldsymbol{x}_j \sim p_\theta(\boldsymbol{x}|\mathcal{F}, \boldsymbol{c}_j)$ for each candidate $\boldsymbol{c}_j$ using iterative sampling
initialize `scores`$[\boldsymbol{c}_j]$ = `list()` for each $\boldsymbol{c}_j$
for trial $t = 1, \dots, T_{\text{sse}}$ **do**
  sample $\sigma \sim \mathcal{N}(0, \sigma_s)$; $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  for candidate $\boldsymbol{c}_k$, sample $\boldsymbol{x}_k$ **do**
    `scores`$[\boldsymbol{c}_k]$.`append`$(\mathcal{L}_{CD}[D_\theta(\boldsymbol{x}_k + \sigma\boldsymbol{\epsilon}, ;\, \mathcal{F}, \emptyset, \sigma), \boldsymbol{x}_k])$
  end for
end for
return $\arg\min_{\boldsymbol{c}_j}$ `mean(scores`$[\boldsymbol{c}_j]$`)`



(a) ATISS [38] **(b)** DiffuScene [51] **(c) DeBaRA**

Figure 3: We compare our method with established baselines for generating a 3D layout from a floor plan and set of object categories. DeBaRA produces less failure cases while consistently generating regular arrangements within the room's bounds.

In practice, we compute an unbiased Monte Carlo estimate of each candidate expectation using T_{sse} fixed $(\sigma, \boldsymbol{\epsilon})$ pairs. Although similar in some aspects, SSE fundamentally differs from diffusion classifiers [22] as in our case, the uniform assumption over conditioning probabilities does not hold. Indeed, in our setting some input signals cannot lead to a plausible arrangement at all. As a result, density estimates of observed samples generated by the class-conditioned model are computed using the unconditional one, while diffusion classifiers compute density estimates of a single observed sample using the class-conditioned model. The SSE procedure is detailed in Algorithm 1. It is further illustrated and discussed in Appendix C.

3.5 Application Scenarios

As shown in Figure 1, a single trained DeBaRA model can be used at test time to perform multiple downstream interactive applications. Usual generation procedures, such as EDM 2$^{\text{nd}}$ order stochastic sampler [19] can be applied using our trained denoiser to generate novel 3D layouts via T-step iterative denoising at discretized noise levels $\sigma_0 = \sigma_{\text{max}} > \ldots > \sigma_T = 0$.

In particular, several applications can be performed by inpainting [30], i.e., predicting missing spatial features from those specified (i.e., fixed) in the input layout $\boldsymbol{x} \in \mathbb{R}^{N \times 8}$. To do so, we introduce a binary mask $\mathbf{m} \in \{0, 1\}^{N \times 8}$ specifying values to retain from the input. The predicted layout at any sampling iteration t can be expressed as:

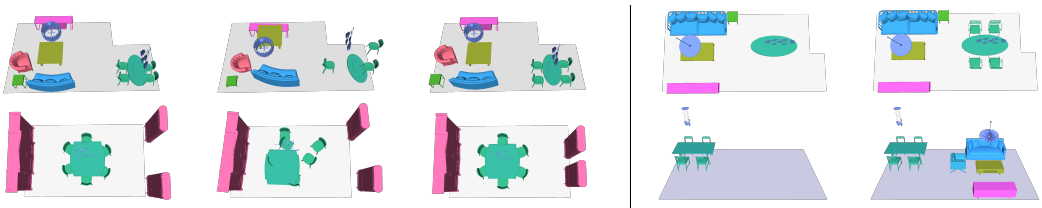
$$\tilde{\boldsymbol{x}}_{\sigma_t} = \hat{\boldsymbol{x}}_{\sigma_t} \odot (1 - \mathbf{m}) + \boldsymbol{x}_{\sigma_t} \odot \mathbf{m} \tag{6}$$

3D Layout Generation Novel and diverse 3D layouts can be generated from an input set of semantic categories $\boldsymbol{c}$ and a floor plan $\mathcal{F}$ by sampling from a high initial noise level $\sigma_{\text{max}} >> \sigma_{\text{data}}$, arbitrarily initialized 3D spatial features $\boldsymbol{x}_{\sigma_0}$ and with $\mathbf{m} = \mathbf{0}_{N \times 8}$.

3D Scene Synthesis DeBaRA can perform 3D scene synthesis via 3D layout generation from semantic categories provided by external sources such as a LLM [9]. Input conditioning candidates can further be optimally selected using the Self Score Evaluation procedure.

Scene Completion Additional objects o_a can be inserted to an existing scene partially furnished with k objects o_e. To do so, their 3D spatial attributes $\boldsymbol{x}_a$ are inpainted from the existing ones $\boldsymbol{x}_e$ with D_θ conditioned on the updated set of semantic categories $\boldsymbol{c} = \boldsymbol{c}_e \| \boldsymbol{c}_a$ using $\mathbf{m}(i, j) = \mathbf{1}_{\{i \leq k\}}$.

Re-arrangement In the context of scene synthesis, re-arrangement [57] consists in recovering the closest *clean* spatial configuration of existing objects from a *messy* one, which has practical applications in robotics [2]. DeBaRA can perform re-arrangement by sampling from an initial noise σ_{max} that depends on the scene perturbation magnitude. During denoising, object positions and rotations $(\boldsymbol{p}, \boldsymbol{r}) \in \mathbb{R}^{N \times 5}$ are inpainted from the known object dimensions using $\mathbf{m}(i, j) = \mathbf{1}_{\{j > 5\}}$.



(a) Ground Truth **(b)** Noisy **(c) Re-arranged** **(d)** Partial **(e) Completed**

Figure 4: Qualitative results on **scene re-arrangement** (left) and **completion** (right). DeBaRA is able to recover a plausible layout from a messy one, and to finely take into account initial configurations.

Optimal Object Retrieval 3D scene synthesis systems depend on external 3D asset databases for furnishing rooms. For each object of semantic class $\boldsymbol{c}$, a textured furniture is retrieved by minimizing the mismatch with the generated dimension $\boldsymbol{d}_{\sigma_T}$. This is inherently suboptimal as the resulting scene quality is limited by the size of the external database. To overcome this issue, we introduce a post-retrieval refinement stage by performing additional *re-arrangement* steps starting from a noise level σ_{max} derived from the mismatch between generated and retrieved object dimensions.

Generation from Coarse Specifications We propose a time-dependent masking approach to synthesize layouts from *rough* input spatial features (i.e., instead of *exact* ones), that are adjusted in the late denoising iterations. To indicate approximate e.g., object dimensions, we set, at any sampling step t, $\mathbf{m}(i,j) = \mathbf{1}_{\{j>5 \text{ and } t<T_s\}}$. The denoising step T_s from which $\mathbf{m}$ is *relaxed* (i.e., set to $\mathbf{0}$) can be derived from its corresponding noise level σ_{T_s} and the precision of specified input features.

4 Experiments

In this section, we provide a comprehensive experimental evaluation of DeBaRA that we compare with established baselines from different model families. We also demonstrate the capabilities of our approach in various practical scenarios, enabling a wide range of applications.

Datasets Our experiments are conducted on the 3D-FRONT [10] synthetic indoor layouts, furnished with assets from 3D-FUTURE [11] that we use as the object retrieval database. Out of the available room types in the dataset, we independently consider living rooms and dining rooms which are more densely furnished and feature complex floor plans. We follow the preprocessing from ATISS [38], leading respectively to 2338/587 and 2071/516 train/test splits.

Baselines We compare DeBaRA with ATISS [38] autoregressive transformer and DiffuScene [51] denoising network. To ensure a fair comparison with our method, we retrained both models with floor plan conditioning on each 3D-FRONT subset using their official implementations. To perform 3D arrangement generation with DiffuScene, we implemented DDPM inpainting [30] of object spatial features from their known semantic categories. Additionally, we report experimental results obtained by LayoutGPT [9] that we implemented with a Llama-3-8B backbone [8] that we also use to provide semantic categories in scene synthesis scenarios. Following the paper, we perform prompting with *supporting examples*: for each test scene, we retrieve top-k samples from the training set that have the most similar floor plan and include their spatial configuration as few-shot exemplars. Note that LayoutGPT adopts a training-free approach and is therefore not directly comparable to our method. However, we show how it can be used *alongside* a specialized model such as DeBaRA. Full implementation details and LLM prompting strategies are reported in Appendix B.3.

Evaluation Metrics We follow previous work [43, 38, 51, 9, 55] and evaluate the realism and diversity of generated arrangements by reporting the 256^2 Fréchet Inception Distance (FID) [15], Kernel Inception Distance (KID $\times 1,000$) [3] and Scene Classification Accuracy (SCA, values closer to 50% are better) computed on top-down orthographic renderings. Resulting projections feature the scene's floor plan and objects colored according to their semantic class [51]. The generation spatial validity is further assessed by reporting the cumulated out of bounds objects area (OBA, in m^2). Related indicators are provided and discussed in Appendix D.1. Metrics are computed across each test subset, for which we generate the same number of scenes as the number of *real* ones.

4.1 3D Layout Generation

The primary task of DeBaRA is to generate diverse and valid 3D layouts within a given floor plan and a list of object semantics. We showcase qualitative generation results and comparisons in Figure 3. As highlighted by previous work [57, 51], denoising-based methods better capture the interplay between interacting objects. We also observe that DeBaRA largely outperforms baselines at respecting the scene's bounds while consistently producing more natural arrangements. These observations are quantitatively verified in Table 1 and visualized in Figure 3.

Table 1: Quantitative experiment results on **bounded 3D layout generation** (providing a floor plan and a list of object semantic categories). We compare our method against other learning-based approaches and additionally indicate results obtained from a training-free LayoutGPT.

Methods	Living Rooms				Dining Rooms			
	FID (↓)	KID (↓)	SCA (%)	OBA (↓)	FID (↓)	KID (↓)	SCA (%)	OBA (↓)
LayoutGPT [9]	35.53	13.69	72.8	2913.6	32.80	8.99	67.6	2447.4
ATISS [38]	25.67	8.91	71.8	857.3	28.05	9.26	63.2	702.4
DiffuScene [51]	21.54	6.40	69.7	341.1	23.06	5.35	57.7	266.4
DeBaRA (ours)	**18.89**	**3.57**	**68.3**	**167.8**	**22.04**	**4.41**	**52.4**	**132.8**

4.2 3D Scene Synthesis and Self Score Evaluation

We demonstrate competitive or state-of-the-art capabilities on 3D scene synthesis against methods that have been specifically trained for this task. We consider several settings depending on the source of input object categories and report our results in Table 2. First, we observe that randomly picking input semantics from the training set (*Dataset Random*) or taking the set $\boldsymbol{c}$ generated by *LayoutGPT* [9] outperform baselines by a significant margin on the 3D-FRONT living rooms test set. Then, to measure the individual impact of SSE, we compare a setup in which input semantics are selected from a set of LLM-generated ones, either randomly (*LLM*) or by applying *SSE*. As LLMs often hallucinate or produce out-of-distribution sets, our procedure consistently improves realism and validity of the synthesized indoor scenes, which can also be qualitatively observed in Figure 5. These results further validate our choice to focus solely on 3D spatial features of objects.

Table 2: Quantitative experiment results on **3D scene synthesis**. DeBaRA is evaluated in various settings based on the source of object semantic categories $\boldsymbol{c}$. Precise settings are detailed and discussed in Appendix B.4. DeBaRA outperforms established baselines on most evaluation metrics.

Methods		Living Rooms				Dining Rooms			
		FID (↓)	KID (↓)	SCA (%)	OBA (↓)	FID (↓)	KID (↓)	SCA (%)	OBA (↓)
LayoutGPT [9]		34.26	10.17	72.1	2902.7	37.78	11.31	60.2	1982.1
ATISS [38]		27.02	10.99	73.0	848.4	28.26	9.28	58.2	759.1
DiffuScene [51]		21.64	5.94	**66.0**	323.1	**23.85**	5.66	54.6	289.8
DeBaRA	*LayoutGPT*	20.97	3.53	69.8	193.0	26.67	7.14	56.6	151.8
	Dataset Random	**19.52**	**3.53**	67.6	**159.0**	25.45	**5.11**	**52.5**	**139.5**
DeBaRA	*LLM*	21.58	3.53	72.4	154.3	27.09	7.38	60.5	140.4
	LLM + SSE	**20.59**	**3.47**	**70.7**	**152.0**	**24.50**	**5.34**	**54.0**	**134.4**

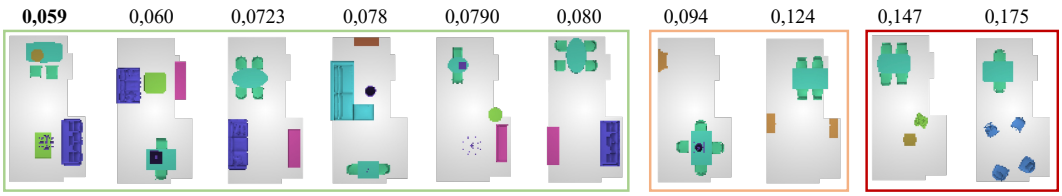
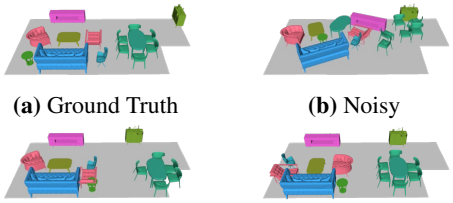


Figure 5: Top-down views of scenes generated by DeBaRA from several *conditioning candidates* provided by a LLM and their associated SSE values. We qualitatively observe that lower scores (*green*) corresponds to more natural layouts while higher scores (*red*) can be filtered out.

4.3 Other Application Scenarios

We present DeBaRA's capabilities at performing additional controllable tasks. Notably, we include quantitative (Table 3) and qualitative (Figure 7) experimental evaluations against LEGO-Net [57] on scene-rearrangement. Results highlight that our method is able to recover more realistic arrangements, while being closer to their initial, *messy* configurations. This is remarkable as the LEGO-Net baseline has been specifically trained to perform this task.



(a) Ground Truth **(b)** Noisy

(c) LEGO-Net [57] **(d)** **DeBaRA**

Figure 7: Qualitative comparison against LEGO-Net [57] on **scene re-arrangement**.

Table 3: Quantitative evaluation on **scene re-arrangement**. DeBaRA is able to recover more realistic arrangements, closer to their initial *noisy* configurations.

Method	FID (↓)	KID (↓)	Distance Moved (↓)
LEGO-Net [57] *grad w/o noise*	26.81	13.18	0.094
DeBaRA	**24.92**	**9.47**	**0.082**

We also provide additional re-arrangement results and showcase DeBaRA's scene completion capabilities, by inserting objects from a list of additional semantics, in Figure 4.

4.4 Ablations

We evaluate the individual contributions of some of our framework's key components on the base 3D layout generation task. Notably, results reported in Table 4 highlight the advantage of our novel objective (Section 3.3) over common formulations as well as the benefits of modeling both the unconditional and class-conditional densities of 3D layouts during training (Section 3.2).

Table 4: Ablation study on DeBaRA **training setup**. We evaluate the individual impact, on 3D layout generation, of different learning objectives $\mathcal{L}$ and of applying conditioning dropout with rate p_{drop}. Notably, the use of our novel Chamfer distance results in a significant performance increase.

Ablation Setting		**Living Rooms**				**Dining Rooms**			
$\mathcal{L}(\hat{\mathcal{O}}, \mathcal{O})$	p_{drop}	FID (↓)	KID (↓)	SCA (%)	OBA (↓)	FID (↓)	KID (↓)	SCA (%)	OBA (↓)
MSE	0.0	21.66	6.55	70.9	237.0	23.89	5.51	56.9	136.5
CD standard	0.0	21.76	7.05	71.7	225.1	25.21	6.75	59.4	294.7
CD semantic-aware **(ours)**	0.0	19.89	4.82	**63.5**	220.0	22.60	4.87	53.4	159.4
CD semantic-aware **(ours)**	0.2	**18.89**	**3.57**	68.3	**167.8**	**22.04**	**4.41**	**52.4**	**132.8**

4.5 Additional Results

Complex Floor Plans We notice that the 3D-FRONT dataset mostly contains *simple* floor maps (i.e., single room, squared, rectangular) both for training and evaluation. As a result, we manually designed irregular floor shapes and report DeBaRA's generation in Figure 8, which further highlights the robustness of our method and its consideration for the conditioning input.

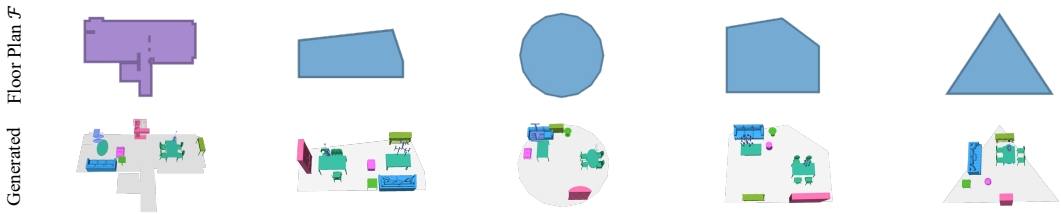


Figure 8: Generated layouts from a given set of objects and **complex floor plans**, selected from the 3D-FRONT test set or handcrafted to irregular, out-of-distribution shapes. While challenging, DeBaRA is able to output plausible layouts in which objects are scattered across the input floor plans.

Iterative Sampling We provide a visualization of the iterative denoising process over time when generating a 3D layout from arbitrarily initialized object bounding boxes in Figure 9.

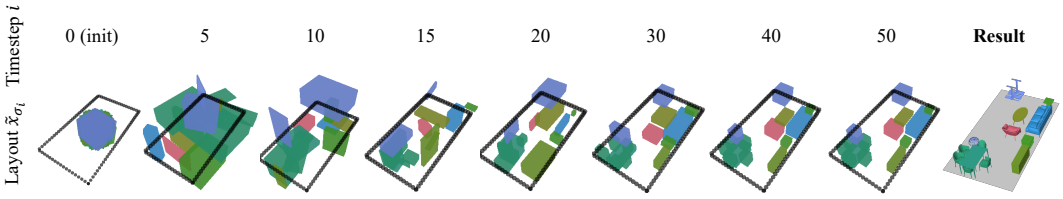


Figure 9: Visualization of intermediate layouts throughout the DeBaRA **denoising process**. *Coarse* object attributes (positions, rotations and dimensions) are determined in the early steps, and then refined in the late iterations.

Generation Variety and Validity We also perform scene completion by adding a bookshelf and a coffee table, repeat the experiment ten times and report in Figure 10 the denoising object trajectories, intermediate and final positions (colored and black dots respectively). This allows to observe the variety of predicted layouts. Notably, we can see that the bookshelf ends up in various different positions, always next to a wall, while the coffee table, which is functionally constrained by its surrounding objects, is placed at similar valid locations.

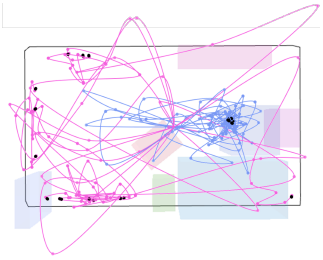


Figure 10: Visualization of sampling trajectories and final positions of a pair of objects, inserted into a scene across ten trials.

Table 5: Generation times are averaged on the 3D-FRONT [10] living room test subset. DeBaRA is implemented with $T = 50$ sampling steps and $T_{\text{sse}} = 100$ Self Score Evaluation trials.

Method	Network Parameters (10^6)	Generation Time (s)
ATISS [38]	36.1	0.160
DiffuScene [51]	89.7	32.796
DeBaRA	12.2	0.488
DeBaRA + SSE	12.2	0.894

Network Efficiency Finally, we compare the number of parameters as well as the sampling (i.e., generation) time, measured on the 3D layout generation task, of our DeBaRA backbone with those of other recent data-driven approaches in Table 5. We can see that our lightweight architecture is bridging the gap with autoregressive methods in terms of inference efficiency.

5 Conclusion, Limitations and Future Work

In this paper we proposed DeBaRA, a novel score-based framework, which achieves state-of-the-art results in 3D layout generation. Our approach is distinctive in its design choices, which both favor data-efficiency with enhanced spatial reasoning, while, at the same time, enabling a range of applications such as scene re-arrangement and completion. Furthermore, we introduce a novel Self Score Evaluation procedure, which allows us to leverage a trained model to select the conditioning signals, which lead to the most plausible results. Overall, our work is the first to unify the conditioning and prediction spaces of score-based models within the context of 3D generative layout.

While powerful, our method currently does not enforce physical constraints between interacting objects, which can lead to collisions. We also assume that object semantic classes are selected among a finite set of predefined categories. Finally, we do not enforce *style consistency* between objects, which can, nevertheless, be performed at retrieval time.

We believe that our approach can enhance other generative models (e.g., architectural layouts, images) by both evaluating the quality and by promoting more plausible 3D layout designs. Furthermore, it will be interesting to combine our approach with encoders from other modalities for a unified multi-modal layout generation.

Acknowledgments and Disclosure of Funding

We thank Despoina Paschalidou for the preprocessing code base and Jiapeng Tang for the mesh rendering pipeline. We also acknowledge the anonymous reviewers for their insightful suggestions. This work was supported by Dassault Systèmes SE. The views and conclusions contained in the paper are those of the authors and should not be interpreted as representing official policies, either expressed or implied, of the company.

References

[1] Armen Avetisyan, Manuel Dahnert, Angela Dai, Manolis Savva, Angel X Chang, and Matthias Nießner. Scan2cad: Learning cad model alignment in rgb-d scans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2614–2623, 2019.

[2] Dhruv Batra, Angel X Chang, Sonia Chernova, Andrew J Davison, Jia Deng, Vladlen Koltun, Sergey Levine, Jitendra Malik, Igor Mordatch, Roozbeh Mottaghi, et al. Rearrangement: A challenge for embodied ai. *arXiv preprint arXiv:2011.01975*, 2020.

[3] Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. *arXiv preprint arXiv:1801.01401*, 2018.

[4] Ryan Burgert, Kanchana Ranasinghe, Xiang Li, and Michael S Ryoo. Peekaboo: Text to image diffusion models are zero-shot segmentors. *arXiv preprint arXiv:2211.13224*, 2022.

[5] Huanran Chen, Yinpeng Dong, Shitong Shao, Zhongkai Hao, Xiao Yang, Hang Su, and Jun Zhu. Your diffusion model is secretly a certifiably robust classifier. *arXiv preprint arXiv:2402.02316*, 2024.

[6] Kevin Clark and Priyank Jaini. Text-to-image diffusion models are zero shot classifiers. *Advances in Neural Information Processing Systems (NeurIPS)*, 36, 2023.

[7] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5828–5839, 2017.

[8] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[9] Weixi Feng, Wanrong Zhu, Tsu-jui Fu, Varun Jampani, Arjun Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. Layoutgpt: Compositional visual planning and generation with large language models. *Advances in Neural Information Processing Systems (NeurIPS)*, 36, 2023.

[10] Huan Fu, Bowen Cai, Lin Gao, Ling-Xiao Zhang, Jiaming Wang, Cao Li, Qixun Zeng, Chengyue Sun, Rongfei Jia, Binqiang Zhao, et al. 3d-front: 3d furnished rooms with layouts and semantics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10933–10942, 2021.

[11] Huan Fu, Rongfei Jia, Lin Gao, Mingming Gong, Binqiang Zhao, Steve Maybank, and Dacheng Tao. 3d-future: 3d furniture shape with texture. *International Journal of Computer Vision (IJCV)*, pages 1–25, 2021.

[12] Rao Fu, Zehao Wen, Zichen Liu, and Srinath Sridhar. Anyhome: Open-vocabulary generation of structured and textured 3d homes. In *European Conference on Computer Vision (ECCV)*, pages 52–70. Springer, 2024.

[13] Lin Gao, Jia-Mu Sun, Kaichun Mo, Yu-Kun Lai, Leonidas J Guibas, and Jie Yang. Scenehgn: Hierarchical graph networks for 3d indoor scene generation with fine-grained geometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 45(7):8902–8919, 2023.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[15] 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. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

[16] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems (NeurIPS)*, 33:6840–6851, 2020.

[17] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[18] Siyuan Huang, Zan Wang, Puhao Li, Baoxiong Jia, Tengyu Liu, Yixin Zhu, Wei Liang, and Song-Chun Zhu. Diffusion-based generation, optimization, and planning in 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16750–16761, 2023.

[19] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:26565–26577, 2022.

[20] Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. Analyzing and improving the training dynamics of diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24174–24184, 2024.

[21] Kurt Leimer, Paul Guerrero, Tomer Weiss, and Przemyslaw Musialski. Layoutenhancer: Generating good indoor layouts from imperfect data. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–8, 2022.

[22] Alexander C Li, Mihir Prabhudesai, Shivam Duggal, Ellis Brown, and Deepak Pathak. Your diffusion model is secretly a zero-shot classifier. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2206–2217, 2023.

[23] Jiahao Li, Hao Tan, Kai Zhang, Zexiang Xu, Fujun Luan, Yinghao Xu, Yicong Hong, Kalyan Sunkavalli, Greg Shakhnarovich, and Sai Bi. Instant3d: Fast text-to-3d with sparse-view generation and large reconstruction model. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

[24] Jiaman Li, Alexander Clegg, Roozbeh Mottaghi, Jiajun Wu, Xavier Puig, and C. Karen Liu. Controllable human-object interaction synthesis. In *European Conference on Computer Vision (ECCV)*, 2024.

[25] Manyi Li, Akshay Gadi Patil, Kai Xu, Siddhartha Chaudhuri, Owais Khan, Ariel Shamir, Changhe Tu, Baoquan Chen, Daniel Cohen-Or, and Hao Zhang. Grains: Generative recursive autoencoders for indoor scenes. *ACM Transactions on Graphics (TOG)*, 38(2):1–16, 2019.

[26] Chenguo Lin and MU Yadong. Instructscene: Instruction-driven 3d indoor scene synthesis with semantic graph prior. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

[27] Jingyu Liu, Wenhan Xiong, Ian Jones, Yixin Nie, Anchit Gupta, and Barlas Oğuz. Clip-layout: Style-consistent indoor scene synthesis with semantic furniture embedding. *arXiv preprint arXiv:2303.03565*, 2023.

[28] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR)*, 2017.

[29] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2018.

[30] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11461–11471, 2022.

[31] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2837–2845, 2021.

[32] Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14297–14306, 2023.

[33] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning (ICML)*, pages 8162–8171. PMLR, 2021.

[34] Alexander Quinn Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob Mcgrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. In *International Conference on Machine Learning (ICML)*, pages 16784–16804. PMLR, 2022.

[35] Yinyu Nie, Angela Dai, Xiaoguang Han, and Matthias Nießner. Learning 3d scene priors with 2d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 792–802, 2023.

[36] 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 (ICCV)*, pages 6690–6700, 2021.

[37] Wamiq Reyaz Para, Paul Guerrero, Niloy Mitra, and Peter Wonka. Cofs: Controllable furniture layout synthesis. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–11, 2023.

[38] Despoina Paschalidou, Amlan Kar, Maria Shugrina, Karsten Kreis, Andreas Geiger, and Sanja Fidler. Atiss: Autoregressive transformers for indoor scene synthesis. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:12013–12026, 2021.

[39] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

[40] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2022.

[41] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 652–660, 2017.

[42] Santhosh Kumar Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alexander Clegg, John M Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X Chang, Manolis Savva, Yili Zhao, and Dhruv Batra. Habitat-matterport 3d dataset (HM3d): 1000 large-scale 3d environments for embodied AI. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

[43] Daniel Ritchie, Kai Wang, and Yu-an Lin. Fast and flexible indoor scene synthesis via deep convolutional generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6182–6190, 2019.

[44] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M. Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *International Conference on Computer Vision (ICCV)*, 2021.

[45] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, 2022.

[46] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:36479–36494, 2022.

[47] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations (ICLR)*, 2022.

[48] Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:1415–1428, 2021.

[49] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.

[50] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016.

[51] 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 (CVPR)*, pages 20507–20518, 2024.

[52] Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, Karsten Kreis, et al. Lion: Latent point diffusion models for 3d shape generation. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:10021–10039, 2022.

[53] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

[54] Kai Wang, Yu-An Lin, Ben Weissmann, Manolis Savva, Angel X Chang, and Daniel Ritchie. Planit: Planning and instantiating indoor scenes with relation graph and spatial prior networks. *ACM Transactions on Graphics (TOG)*, 38(4):1–15, 2019.

[55] Xinpeng Wang, Chandan Yeshwanth, and Matthias Nießner. Sceneformer: Indoor scene generation with transformers. In *International Conference on 3D Vision (3DV)*, pages 106–115. IEEE, 2021.

[56] Daniel Watson, William Chan, Ricardo Martin Brualla, Jonathan Ho, Andrea Tagliasacchi, and Mohammad Norouzi. Novel view synthesis with diffusion models. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2022.

[57] Qiuhong Anna Wei, Sijie Ding, Jeong Joon Park, Rahul Sajnani, Adrien Poulenard, Srinath Sridhar, and Leonidas Guibas. Lego-net: Learning regular rearrangements of objects in rooms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19037–19047, 2023.

[58] Zhennan Wu, Yang Li, Han Yan, Taizhang Shang, Weixuan Sun, Senbo Wang, Ruikai Cui, Weizhe Liu, Hiroyuki Sato, Hongdong Li, et al. Blockfusion: Expandable 3d scene generation using latent tri-plane extrapolation. *ACM Transactions on Graphics (TOG)*, 43(4):1–17, 2024.

[59] Xiang Xu, Joseph Lambourne, Pradeep Jayaraman, Zhengqing Wang, Karl Willis, and Yasutaka Furukawa. Brepgen: A b-rep generative diffusion model with structured latent geometry. *ACM Transactions on Graphics (TOG)*, 43(4):1–14, 2024.

[60] Yandan Yang, Baoxiong Jia, Peiyuan Zhi, and Siyuan Huang. Physcene: Physically interactable 3d scene synthesis for embodied ai. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16262–16272, 2024.

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

[62] 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 graphs. *Advances in Neural Information Processing Systems (NeurIPS)*, 36, 2023.

[63] Genghao Zhang, Yuxi Wang, Chuanchen Luo, Shibiao Xu, Junran Peng, Zhaoxiang Zhang, and Man Zhang. Furniscene: A large-scale 3d room dataset with intricate furnishing scenes. *arXiv preprint arXiv:2401.03470*, 2024.

[64] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3836–3847, 2023.

[65] Zaiwei Zhang, Zhenpei Yang, Chongyang Ma, Linjie Luo, Alexander Huth, Etienne Vouga, and Qixing Huang. Deep generative modeling for scene synthesis via hybrid representations. *ACM Transactions on Graphics (TOG)*, 39(2):1–21, 2020.

[66] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5745–5753, 2019.

Appendix

A Score-based Framework

In this section, we give additional details on the score-based parameterization that we adopt to learn the distribution of 3D layouts $p_{\text{data}}(\boldsymbol{x})$ and the sampling strategy used to generate new samples from the resulting trained denoiser D_θ.

A.1 Training

Score-based approaches model the score (i.e., the gradient of log-probability density w.r.t. the data) of marginal distributions $p_\sigma(\boldsymbol{x})$ obtained by perturbing the data with Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ at magnitudes σ. In practice, the score can be effectively approximated by a noise-conditioned denoiser that outputs clean samples from noisy ones, then, $\nabla_{\boldsymbol{x}} \log p_\sigma(\boldsymbol{x}) = (D_\theta(\boldsymbol{x}; \boldsymbol{y}, \sigma) - \boldsymbol{x})/\sigma^2$.

Parameterizing the denoiser to output $\boldsymbol{x}$ from its corrupted version directly is not ideal as the input magnitude varies greatly depending on the current noise level. Instead, Karras et al. [19] propose in their EDM diffusion framework a preconditioning of the denoiser whose output is now derived from a trainable network F_θ that either predicts the clean signal $\boldsymbol{x}$, the noise $\boldsymbol{\epsilon}$ or something in between, depending on the value of σ. More formally, it can be expressed as:

$$D_\theta(\boldsymbol{x}; \boldsymbol{y}, \sigma) = c_{\text{skip}}(\sigma)\, \boldsymbol{x} + c_{\text{out}}(\sigma)\, F_\theta\big(c_{\text{in}}(\sigma)\, \boldsymbol{x}; \boldsymbol{y}, c_{\text{noise}}(\sigma)\big) \tag{7}$$

The preconditioning function c_{skip} amplifies the network error as little as possible while c_{in} and c_{out} scale respectively the input and output to have unit variance. Following [19], we set:

$$c_{\text{skip}} = \frac{\sigma_{\text{data}}^2}{\sigma_{\text{data}}^2 + \sigma^2}\ ;\ c_{\text{in}} = \frac{1}{\sqrt{\sigma_{\text{data}}^2 + \sigma^2}}\ ;\ c_{\text{out}} = \frac{\sigma \cdot \sigma_{\text{data}}}{\sqrt{\sigma_{\text{data}}^2 + \sigma^2}}\ ;\ c_{\text{noise}} = \frac{\ln(\sigma)}{4} \tag{8}$$

Note that in our case, the value of σ_{data} should preferably be computed channel-wise for each attribute of $\boldsymbol{x} = (\boldsymbol{p}, \boldsymbol{r}, \boldsymbol{d}) \in \mathbb{R}^8$, as object positions, rotations and dimensions typically have different standard deviations. In practice, we compute $\sigma_{\text{data}}^{\boldsymbol{p}}$ and $\sigma_{\text{data}}^{\boldsymbol{d}}$ from the training data and arbitrarily set $\sigma_{\text{data}}^{\boldsymbol{r}} = (0.5, 0.5)$. During training, noise values σ are drawn from a centered normal distribution of variance 0.25, which concentrates training on *medium* noise levels. Spatial values $\boldsymbol{p}$ and $\boldsymbol{d}$ of each training layout $\boldsymbol{x}$ are normalized based on the maximum extent of the scene's floor plan $\mathcal{F}$, which ensures that all the network's inputs and outputs are scaled in $[-1, 1]$. To model both the class-conditional and unconditional layout densities, we perform conditioning dropout on object categories $\boldsymbol{c}$ with a rate $p_{\text{drop}} = 0.2$.

Finally, the training objective can be expressed by introducing our semantic-aware Chamfer reconstruction loss following Equation 3. As in EDM, we use $\lambda(\sigma) = 1/c_{\text{out}}^2$ to get a uniform weighting across noise levels.

A.2 Sampling

At test time, the reverse SDE [19, 49] associated to the continuous-time diffusion process is used to generate novel samples from a standard normal distribution using numerical solvers. It depends on the score approximated during training:

$$d\boldsymbol{x} = \big[f(\boldsymbol{x}, t) - g(t)^2 \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})\big] dt + g(t) dw \tag{9}$$

where $f(\cdot, t) : \mathbb{R}^d \to \mathbb{R}^d$ and $g(\cdot) : \mathbb{R} \to \mathbb{R}$ are respectively the drift and diffusion coefficients and w is the standard stochastic Wiener process.

In practice, we use EDM [19] 2$^{\text{nd}}$ order Runge-Kutta stochastic sampler (see Algorithm 2), that resembles the predictor/corrector framework from Song et al. [49] and provides a good trade-off between generation quality and number of function evaluations (NFE).

It is based on a T-step discretization of the reverse SDE [19, 49], with timesteps $t_{i \in \{0, \dots, T-1\}}$ decreasing from σ_{max} ($i = 0$) to σ_{min} ($i = T - 1$). Following [19], we use:

$$t_{i<T} = \left(\sigma_{\text{max}}{}^{\frac{1}{\rho}} + \frac{i}{T-1}\left(\sigma_{\text{min}}{}^{\frac{1}{\rho}} - \sigma_{\text{max}}{}^{\frac{1}{\rho}}\right)\right)^{\rho}, \quad t_T = 0 \tag{10}$$

Algorithm 2 EDM Stochastic Sampler [19]

procedure LAYOUTSAMPLER($D_\theta(\boldsymbol{x}; \boldsymbol{y}, \sigma), \boldsymbol{x}_0, t_{i\in\{0,\dots,T-1\}}, \gamma_{i\in\{0,\dots,T-1\}}, S_{\text{noise}}$)
  for $i \in \{0, \dots, T-1\}$ **do**
    sample $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, S_{\text{noise}}^2\mathbf{I})$
    $\hat{t}_i \leftarrow t_i + \gamma_i t_i$
    $\hat{\boldsymbol{x}}_i \leftarrow \boldsymbol{x}_i + \sqrt{\hat{t}_i^2 - t_i^2}\boldsymbol{\epsilon}_i$
    $\boldsymbol{g}_i \leftarrow \left(\hat{\boldsymbol{x}}_i - D_\theta(\hat{\boldsymbol{x}}_i; \boldsymbol{y}, \hat{t}_i)\right)/\hat{t}_i$
    $\boldsymbol{x}_{i+1} \leftarrow \hat{\boldsymbol{x}}_i + (t_{i+1} - \hat{t}_i)\boldsymbol{g}_i$
    if $t_{i+1} \neq 0$ **then**
      $\boldsymbol{g}'_i \leftarrow \left(\boldsymbol{x}_{i+1} - D_\theta(\boldsymbol{x}_{i+1}; \boldsymbol{y}, t_{i+1})\right)/t_{i+1}$
      $\boldsymbol{x}_{i+1} \leftarrow \hat{\boldsymbol{x}}_i + \frac{1}{2}(t_{i+1} - \hat{t}_i)(\boldsymbol{g}_i + \boldsymbol{g}'_i)$
    end if
  end for
end procedure

Note that t and noise level σ can be used interchangeably. The ρ parameter is tuned to dedicate more steps of the denoising process to smaller or larger noise levels. The σ_{min} value should be small enough so that the model estimates the best approximation of the score and sample a precise layout. On the other hand, σ_{max} should be large enough to sample various layouts. The amount of *fresh* noise injected at the beginning of each denoising step is defined by $\gamma_{i\in\{0,\dots,N-1\}}$. Similar to Wei et al. [57] and Karras et al. [19], we qualitatively observed that adding noise in the final timesteps, i.e., when the layout is close to its final configuration, leads to less precise results. As a result, an additional S_{min} parameter is set so that $\gamma_i = 0$ when $t_i < S_{\text{min}}$.

For 3D layout generation, we use $T = 50$ timesteps and set $\sigma_{\text{max}} = 1.0$, $\sigma_{\text{min}} = 0.005$, $\rho = 7$, and $S_{\text{min}} = 0.005$. Note that, although we perform conditioning dropout during training, we didn't find the need to amplify the strength of the input categories using any classifier-free guidance [17] scale at sampling time.

Additionally, Table 6 shows that sampling from DeBaRA using EDM [19] 2$^{\text{nd}}$ order stochastic procedure (Algorithm 2) outperforms ancestral DDPM [16] sampling using a fraction of the denoising steps. This, combined with our lightweight architecture, enables *real-time* (<1s) generation.

Table 6: Ablation study on DeBaRA **sampling strategy**. Metrics are computed on the 3D layout generation task.

Sampler		Living Rooms					Dining Rooms				
Alg.	Steps	FID (↓)	KID (↓)	SCA (%)	OBA (↓)	Time (s)	FID (↓)	KID (↓)	SCA (%)	OBA (↓)	Time (s)
DDPM	1000	21.12	5.65	**67.4**	268.9	5.144	23.18	5.78	53.3	202.9	4.925
EDM	25	19.53	3.95	69.4	**159.5**	**0.247**	**21.95**	**4.26**	54.7	140.5	**0.248**
EDM	50	**18.89**	**3.57**	68.3	167.8	0.488	22.04	4.41	**52.4**	**132.8**	0.514

B Implementation

We provide in this section additional implementation details on our model architecture and training configurations, illustrated in Figure 2. We also detail how baselines have been retrained and used at test time to ensure a fair and relevant comparison with our approach.

B.1 Network Architecture

Shared Object Encoder The shared object encoder embeds each object o_i from its input 3D spatial values $\boldsymbol{x}_i$ and semantic category $\boldsymbol{c}_i$. Triplets of scalar values of the object's position $\boldsymbol{p}_i$ and dimension $\boldsymbol{d}_i$ are encoded with fixed sinusoidal positional encoding of 32 frequencies following [57]:

$$PE(s) = \left\{\sin(128^{j/31}s), \cos(128^{j/31}s)\right\}_{j=0}^{31} \in \mathbb{R}^{64} \tag{11}$$

Applying this module projects $\boldsymbol{p}_i$ and $\boldsymbol{r}_i$ in $\mathbb{R}^{192}$. It is similarly applied to $\boldsymbol{r}_i = (\cos(\theta_i), \sin(\theta_i))$ to get a feature in $\mathbb{R}^{128}$, that is additionally fed to a linear layer to obtain a 192-dimensional attribute.

The object semantic class $\boldsymbol{c}_i$, represented as a one-hot vector among k classes is encoded in $\mathbb{R}^{192}$ by an MLP with 2 a hidden layer of 128 units and LeakyReLU activation. Respective object spatial and semantic encodings are then concatenated to form an object token $\mathcal{T}_{o_i}$ of dimension $4 \times 192 = 768$.

Floor Encoder The scene's conditioning floor plan $\mathcal{F}$ is embedded by a PointNet [41] module, similar to [57]. To do so, we first extract the floor's 2D polygon using the output of ATISS [38] preprocessing and sample $P = 100$ evenly spaced points on its contour. The PointNet backbone[2] produces a 1024-dimensional feature, that is further passed to a linear layer to get the appropriate floor token $\mathcal{T}_{\mathcal{F}} \in \mathbb{R}^{768}$.

Noise Level Encoder We encode the noise level σ as a token $\mathcal{T}_{\sigma} \in \mathbb{R}^{768}$ obtained by subsequently applying $PE(\sigma)$ and a linear layer with LeakyReLU activation.

Transformer Encoder Our transformer encoder that computes new representations $\hat{\mathcal{T}}$ is composed of multi-head self-attention and feedforward layers, following the original paper [53] and implementation from the PyTorch [39] API. Importantly, we don't enforce ordering of any input token and pass an additional padding mask to handle sequences of different lengths. We stack 3 encoder layers, each having 4 attention heads and a feedforward hidden dimension of 512.

Shared Object Decoder The final shared object decoder produces the network's predicted spatial values $\hat{\boldsymbol{x}}_i \in \mathbb{R}^8$ for each of the N objects from their respective $\hat{\mathcal{T}}_{o_i}$ embeddings. It is implemented as an MLP with three layers of 512, 128, and 8 units, using LeakyReLU activations and a dropout rate of 0.1.

B.2 Training Protocol

During training, the network is optimized towards our semantic-aware Chamfer loss, that can be efficiently implemented with appropriate broadcasting. We trained our models separately on the 3D-FRONT [10] *living room* and *dining room* subsets for 3000 epochs, with a batch size of 32 and monitor the validation loss to avoid overfitting of the training set in the late iterations. We use the AdamW [29] optimizer with its PyTorch default parameters and learning rate $\eta = 10^{-4}$, scheduled with a linear warmup phase for the first 50 epochs, starting at $\eta \times 0.01$. Following this, a cosine annealing schedule [28] reduces η to a minimum of 10^{-8} over 2200 epochs. Finally, we randomly apply rotations to the training scenes as data augmentation.

B.3 Baselines

ATISS ATISS [38] is an autoregressive, permutation-invariant transformer that treats 3D scene synthesis as an unordered set generation task. The model is natively conditioned on the room's floor plan, from which it extracts features using a ResNet-18 [14] applied on a top-down binary projection. The model predicts the semantic class, location, rotation and dimension of the next object to be inserted to the current layout configuration. As our method, it also supports inserting objects from their semantic categories given as input, which is the setting that we used to report experimental results on the 3D layout generation task (see Table 1 and Figure 3). We retrained the model on each 3D-FRONT subset using the authors' implementation.[3]

DiffuScene DiffuScene [51] employs a DDPM to perform 3D scene synthesis, by learning to denoise unordered sets of objects that are each represented by all their attributes, i.e., location, size, orientation, semantic category and *shape code*. Although the paper specifically mentions not being conditioned on the room's bounds, we found out the official implementation supports this feature that we enabled to retrain the model on the 3D-FRONT subsets, with other settings set to those of the authors. In practice and similar to ATISS [38], a ResNet-18 backbone is used to extract features from the floor plan's projection mask. The resulting encoding is passed to an MLP whose output is added to the diffusion timestep embedding, as in [46]. To assess the effectiveness of this conditioning mechanism and validate the relevance of this baseline, we report metrics obtained for the 3D layout

[2] https://github.com/fxia22/pointnet.pytorch
[3] https://github.com/nv-tlabs/ATISS

generation task, using both a floor-conditioned and an unconditional (*vanilla*) trained DiffuScene model on the *living room* subset in Table 7. Note that, top-down view of scenes, that we use to evaluate e.g., FID and KID scores, feature the floor plan, which penalizes generated configurations that don't properly take it into account.

Table 7: Quantitative impact of DiffuScene [51] floor plan conditioning on 3D layout generation. Results indicate that the additional input has successfully been learned, resulting in a solid baseline.

Model	**FID** (↓)	**KID** (↓)	**OBA** (m^2)
DiffuScene *vanilla*	41.30	22.92	1621.5
DiffuScene *floor*	**21.54**	**6.40**	**341.1**

To perform 3D layout generation from input semantic categories using DiffuScene (Table 1, Figure 3), we implemented DDPM inpainting of the object spatial features from their categories as an additional method within the official implementation[4] and using default sampling settings with 1000 timesteps.

LayoutGPT LayoutGPT [9] is a training-free approach that utilizes Large Language Models to generate layouts both in the image and the 3D scene domains, demonstrating competitive performance with learning-based approaches on 3D scene synthesis. To do so, the method consists in prompting a LLM with specific instructions and by adding *supporting examples* from the training set, i.e., few-shot exemplars of expected, valid layouts. These examples are retrieved from the train set based on floor plan similarity computed from the binary masks with a test sample. We reimplemented LayoutGPT, using the official implementation[5] for exact prompt and *supporting examples* retrieval, but using a Meta Llama-3-8B[6] backbone instead of ChatGPT variants for local execution and better reproducibility.

To perform 3D layout generation, we include in the prompt the list of object semantic categories. Here is a typical LayoutGPT prompt for this task on the *living room* subset:

```
Instruction: synthesize the 3D layout of an indoor scene. The generated 3D
layout should follow the CSS style, where each line starts with the furniture
category and is followed by the 3D size, orientation and absolute position.
Formally, each line should follow the template:
FURNITURE {length: ?m: width: ?m; height: ?m; left: ?m; top: ?m; depth: ?m;
orientation: ? degrees;}
All values are in meters but the orientation angle is in degrees.

Condition:
Room Type: livingroom
Room Size: max length 1.00m, max width 1.53m
Furniture categories: console_table multi_seat_sofa corner_side_table
corner_side_table coffee_table tv_stand pendant_lamp
Layout:
console_table {length: 0.28m; height: 0.31m; width: 0.14m; orientation: -90
degrees; left: 0.93m; top: 0.41m; depth: 0.15m;}
[…]

[OTHER SUPPORTING EXAMPLES]

Condition:
Room Type: livingroom
Room Size: max length 1.00m, max width 1.62m
Furniture categories: tv_stand corner_side_table coffee_table dining_table
dining_chair dining_chair dining_chair dining_chair armchair pendant_lamp
pendant_lamp multi_seat_sofa
Layout:
```

For 3D scene synthesis, we follow the paper and include the training set's object frequencies in the prompt. We set the LLM sampling temperature to 0.7 and maximum output tokens to 1024. We

[4] https://github.com/tangjiapeng/DiffuScene
[5] https://github.com/weixi-feng/LayoutGPT
[6] https://huggingface.co/meta-llama/Meta-Llama-3-8B

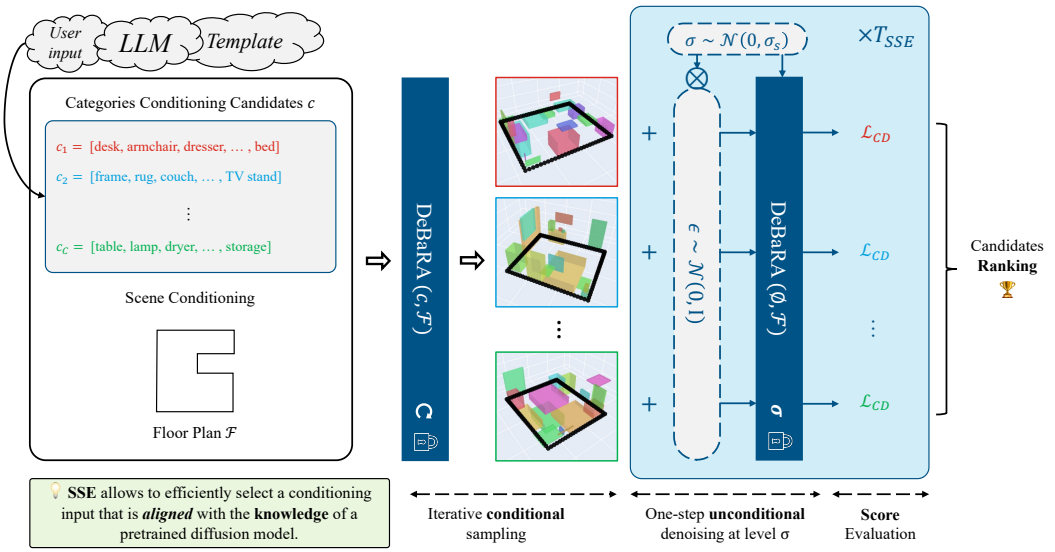


Figure 11: Self Score Evaluation (SSE) Pipeline. The method allows to leverage the knowledge of a model trained following our method to select valid sets of object categories.

report LayoutGPT performance on this task in Table 2, where we also indicate metrics obtained by DeBaRA when using as conditioning input the same semantic set as the one generated by LayoutGPT for the corresponding test scene. In this setup, we largely outperform the baseline and even report state-of-the-art FID, KID and OBA scores on 3D-FRONT *living room*.

LEGO-Net The LEGO-Net [57] model is specifically designed to perform 2D scene re-arrangement, i.e., recover a close *clean* layout configuration from a *messy*, perturbed one. It is trained using a regression loss on object position and rotation values, and proposes a Langevin dynamics-like iterative sampling procedure. In order to produce the results reported in Table 3, we used authors' implementation,[7] in the *grad without noise* setting (which is the best performing in the original paper), on the 3D-FRONT *living room* test subset and with a scene perturbation level of 0.25.

B.4 LLM Prompting

As mentioned in the main submission, we perform 3D scene synthesis using DeBaRA conditioned on LLM-generated sets of object semantics, that we optionally select via *Self Score Evaluation* (Table 2, Figure 5). In practice, the generated categories have been obtained using Llama-3-8B [8] following the LayoutGPT [9] prompting strategy. We experienced asking the language model to generate solely lists of object semantics using a few supporting examples and providing the dataset statistics, but we noticed that it was more prone to hallucinate and drift towards inconsistent generations than when generating complete layout configurations (i.e., including object position and orientation values).

The *DeBaRA LLM* reported in Table 2 corresponds to the setup where we filter from the LLM-generated sets of categories those having the same number of objects as the considered test scene and randomly select one to condition DeBaRA. The *DeBaRA LLM + SSE* setting is similar, but instead of picking a set randomly, the selection is performed by applying the SSE procedure.

B.5 Computational Requirements

All the training and evaluation experiments as well as the computation of generation times reported in Table 5 have been performed on a single NVIDIA RTX A6000 GPU. When comparing our number of network parameters and generation times with those of ATISS [38] and DiffuScene [51], we notice that DeBaRA is bridging the gap with autoregressive methods in terms of inference efficiency. This is made possible by our restricted output space that requires a more lightweight backbone as well as our choice of sampling procedure that leads to a favorable NFE / generation quality tradeoff.

[7] https://github.com/QiuhongAnnaWei/LEGO-Net

C Self Score Evaluation

In this section, we provide additional content regarding our SSE procedure (Section 3.4), by illustrating it and by further assessing its expressive power.

C.1 Pipeline

The SSE formulation is expressed by Equations 4 - 5. It follows the procedure outlined in Algorithm 1. We additionally illustrate the SSE pipeline in Figure 11. In our experiments, SSE is implemented using $T_{\text{sse}} = 100$ trials, with noise levels σ drawn as in training.

C.2 Additional Evaluation

We additionally evaluate the expressive power of the SSE procedure in a *toy* binary classification task: for each scene of the test set, we create a corrupted version by replacing a proportion p_{rand} of the scene's object categories by random ones. We report the binary classification score obtained by SSE when asked to discriminate the corrupted set of semantics from the ground truth one. We perform the experiments 10 times to account for the inherent stochasticity of the experimental setup and report the results for several values of p_{rand} in Table 8.

Table 8: SSE performance at discriminating ground truth sets of categories from perturbed ones.

Perturbation	None	Single	$p_{\text{rand}} = 0.35$	$p_{\text{rand}} = 0.50$	$p_{\text{rand}} = 0.75$	All
Accuracy (%)	50.6 ± 4.39	71.7 ± 2.20	73.8 ± 3.38	78.5 ± 1.84	80.9 ± 2.62	84.0 ± 2.90

We observe a significant gap in accuracy between the control experiment (none object is perturbed, meaning that the sets are equals) and the setting where only a single object has been swapped. It means that SSE is able to identify subtle misalignments between the network's knowledge and the provided conditioning candidates.

D Additional Results

In this section, we provide supplementary quantitative indicators measuring the validity of generated layouts with respect to the conditioning floor plan. We also showcase additional qualitative results and comparisons in multiple application scenarios.

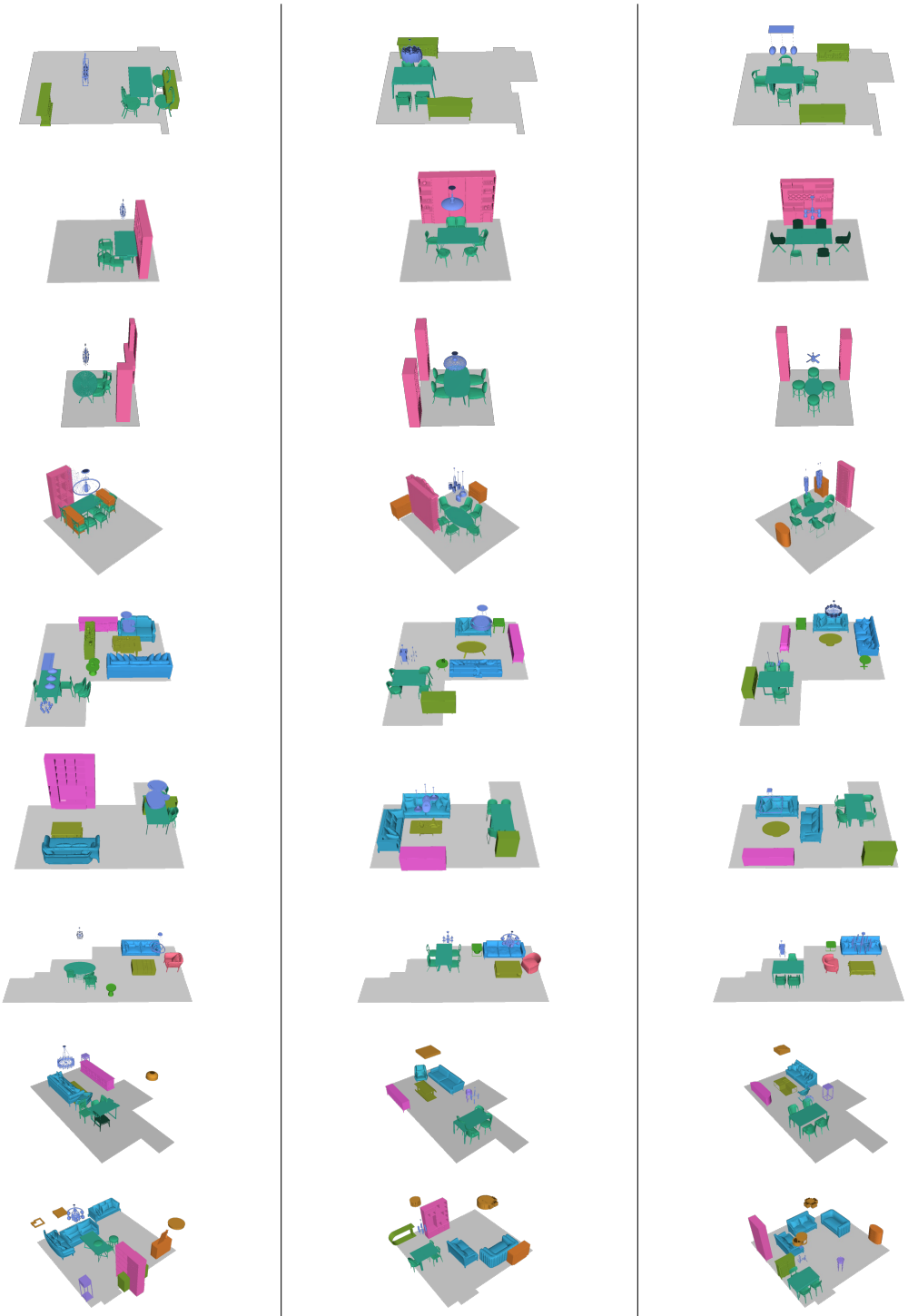
D.1 Bounding Metrics

In addition to the cumulated out of bounds objects area (OBA, in m^2) reported in Table 1, we indicate the rate of scenes having at least one object out of its bounds (OBR), and the cumulated number of out-of-bounds objects in the generated layouts (OBN) in Table 9. For OBR and OBN, we consider an object to be out-of-bounds if at least 20% of its 2D bounding box surface is outside the floor's limits.

Table 9: Quantitative bounding metrics on the 3D layout generation task. We observe that DeBaRA is consistently better at respecting the indoor floor plan by a significant margin.

Methods	**Living Rooms**			**Dining Rooms**		
	OBA (↓)	OBR (↓)	OBN (↓)	OBA (↓)	OBR (↓)	OBN (↓)
LayoutGPT [9]	2913.6	0.695	2119	2447.4	0.659	1720
ATISS [38]	857.3	0.744	1195	702.4	0.891	1603
DiffuScene [51]	341.1	0.652	742	266.4	0.628	640
DeBaRA (ours)	**167.8**	**0.390**	**497**	**132.8**	**0.403**	**401**

D.2 Qualitative Results

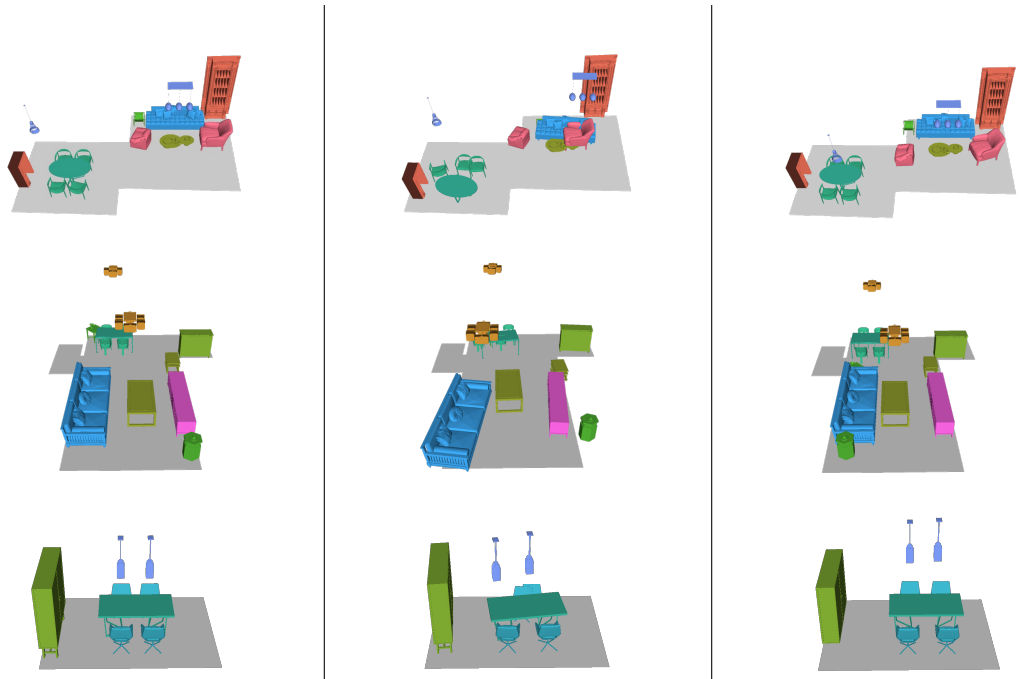


(a) ATISS [38] **(b)** DiffuScene [51] **(c)** **DeBaRA**

Figure 12: Additional 3D layout generation results. We compare our method by generating layouts from a list of object semantic categories and a floor plan. DeBaRA consistently produces more realistic arrangements while respecting the room's outline.

(a) ATISS [38] | **(b)** DiffuScene [51] | **(c) DeBaRA** *Dataset Random* | **(d) DeBaRA** *LayoutGPT* | **(e) DeBaRA** *SSE*

Figure 13: Additional scene synthesis results. We compare DeBaRA with state-of-the-art approaches in various settings. Since object semantics are not part of our output space, they are randomly drawn from the training dataset (*Dataset Random*), generated by an external LLM (*LayoutGPT*) or selected from LLM-generated sets by our density estimate procedure (*SSE*), which helps to get more natural layouts for the considered floor configuration.



(a) Ground Truth **(b)** Perturbed **(c) Re-arranged**

Figure 14: Additional re-arrangement results. Our method can recover a plausible arrangement from a noisy one using iterative sampling with fixed dimension attributes and low initial noise level.

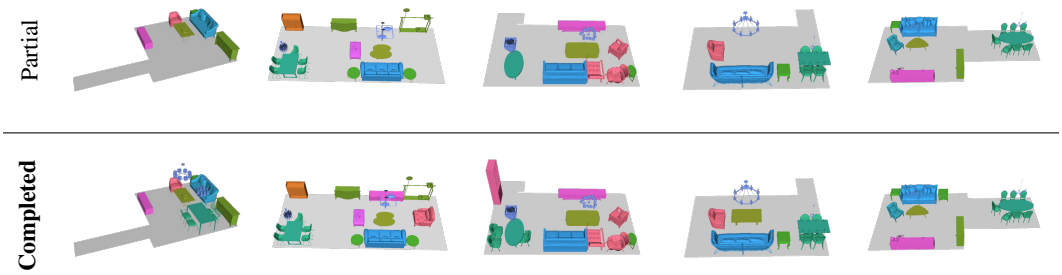
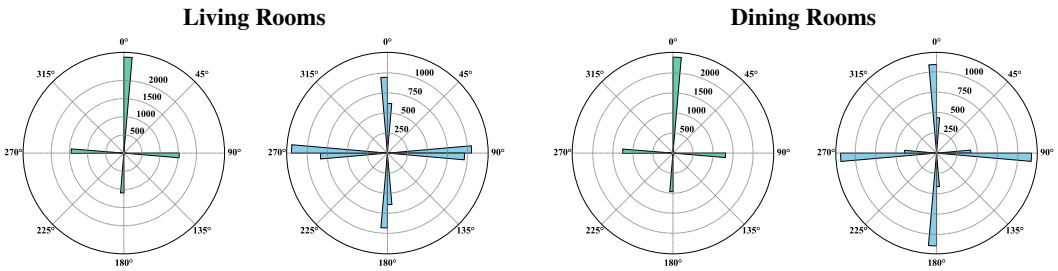


Figure 15: Additional scene completion results. From a list of additional objects semantics, DeBaRA is able to finely introduce the relevant items into an existing layout.

D.3 Statistical Analysis

We can observe in our qualitative results, e.g., in Figure 3, that objects are most of the time parallel or perpendicular to walls, meaning that their rotation attributes $\boldsymbol{r}$ mainly take *cardinal* angle values, i.e., 0°, 90°, 180° or 270°.

Radial histograms of object rotation values reported in Figure 16 indicate that 3D-FRONT [10] subsets contain rooms in which objects, in their vast majority, exhibit this property and don't have much *exotic* angles, i.e., different from cardinal ones. These restricted rotation distributions are consequently learned and reflected in the outputs of generative models at sampling time.



(a) Dataset **(b)** DeBaRA **(c)** Dataset **(d)** DeBaRA

Figure 16: Distributions of object rotation values extracted from the dataset or generated following our method, reported for both test subsets. We can see that the original data predominantly features objects rotated by 0°, 90°, 180° or 270° around the scene's vertical axis, which, as a result, is also the tendency induced by the generative model.

E Societal Impact

We believe that our method will predominantly yield positive societal impact by enhancing accessibility, controllability and efficiency in 3D indoor design and creation, benefiting both non-experts and professionals in several industries.

Nevertheless, it raises common concerns associated with deep generative models. Specifically, users should be warned about using the model for furnishing real-world rooms as generated layouts may result in unsafe designs. Additionally, privacy safeguards should systematically be implemented along our method in order to prevent unauthorized replication of personal spaces.

F Licenses

In this section, we list licenses of datasets, open-source code artifacts and pretrained models that were used in the context of our experiments.

Datasets

- 3D-FRONT [10]: CC BY-NC-SA 4.0 License
- 3D-FUTURE [11]: CC BY-NC-SA 4.0 License

Code

- ATISS [38]: NVIDIA Source Code License for ATISS[8]
- DiffuScene [51]: Sony Group Corporation License for DiffuScene[9]
- LayoutGPT [9]: MIT License
- LEGO-Net [57]: MIT License
- PointNet [41]: MIT License

Models

- Llama-3-8B [8]: Meta Llama 3 Community License Agreement[10]
- Inception-v3 [50]: Apache 2.0 License

[8] https://github.com/nv-tlabs/ATISS/blob/master/LICENSE
[9] https://github.com/tangjiapeng/DiffuScene/blob/master/LICENSE
[10] https://huggingface.co/meta-llama/Meta-Llama-3-8B/blob/main/LICENSE

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: Claims made in the abstract are all developed throughout the paper, notably in the *method* Section 3.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We discuss the paper's main limitations in the last Section 5, along with the conclusion.

   Guidelines:

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

3. **Theory Assumptions and Proofs**

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

Answer: [Yes]

Justification: For our main theoretical contribution, Self Score Evaluation (Section 3.4), we clearly state that it is made possible under the assumption that the denoiser models both the unconditional and the class-conditional data distributions.

Guidelines:

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

4. **Experimental Result Reproducibility**

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

Answer: [Yes]

Justification: We comprehensively detail our experimental setup, both for our method and the baselines in the main paper, Section 4, and in Appendix A and B.

Guidelines:

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

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

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

   Answer: [No]

   Justification: Unfortunately, the code for DeBaRA cannot be disclosed due to author affiliation. However, we comprehensively describe our settings and implementation details in Appendix A and B and the 3D-FRONT [10] dataset is openly available.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

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

   Answer: [Yes]

   Justification: All the relevant details are given in the paper, notably in Section 4, as well as in Appendix B.

   Guidelines:

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

7. **Experiment Statistical Significance**

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

   Answer: [No]

   Justification: Unfortunately, we couldn't perform such analysis due to limited computational resources.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

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

   Answer: [Yes]

   Justification: We indicate our GPU setting in Appendix B.5 and report network execution time in the main submission (Table 5).

   Guidelines:

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

9. **Code Of Ethics**

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

   Answer: [Yes]

   Justification: The research conducted in the paper conforms, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader Impacts**

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

    Answer: [Yes]

    Justification: Potential societal impacts, both positive or negative, are discussed in a dedicated section in Appendix E. We also discuss possible ways to mitigate potential negative societal impacts.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: Creators or owners of assets are either credited via citation of their work (e.g., 3D-FRONT [10] and 3D-FUTURE [11] datasets) or direct link to their code or model (throughout Appendix B). A dedicated section in Appendix F explicitly mentions the relevant licenses, which have been properly respected to conduct our work.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

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

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

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

14. **Crowdsourcing and Research with Human Subjects**

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

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

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

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.